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.

Студент гр. 0382 _____ Кривенцова Л.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить принцип связи программ на Ассемблере с кодом на языках программирования высокого уровня. Реализовать программу, решающую задачу с помощью взаимосвязи языков программирования C++ 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.

Вариант 9.

№	Вид распредения	Число ассем. процедур	Nint ≥ D_x	Nint ≤ D_x	Lgi ≤ X_{min}	Lg1 > X_{min}	ПГпосл ≤ X_{max}	ПГпосл ≥ X_{max}
9	равном.	1	-	+	-	+	-	+

Основные теоретические положения.

Существуют следующие формы комбинирования программ на языках высокого уровня с ассемблером:

- Использование ассемблерных вставок (встроенный ассемблер, режим inline). Ассемблерные коды в виде команд ассемблера вставляются в

текст программы на языке высокого уровня. Компилятор языка распознает их как команды ассемблера и без изменений включает в формируемый им объектный код. Эта форма удобна, если надо вставить небольшой фрагмент.

- Использование внешних процедур и функций. Это более универсальная форма комбинирования. У нее есть ряд преимуществ:
  — написание и отладку программ можно производить независимо;
  — написанные подпрограммы можно использовать в других проектах;
  — облегчаются модификация и сопровождение подпрограмм.
  Использование внешних процедур.

Для связи посредством внешних процедур создается многофайловая программа. При этом в общем случае возможны два варианта вызова:

- программа на языке высокого уровня вызывает процедуру на языке ассемблера;
- программа на языке ассемблера вызывает процедуру на языке высокого уровня.

В программах, написанных на языке ассемблера, используется соглашение передачи параметров stdcall. Однако по сути получение и передача параметров в языке ассемблера производится явно, без помощи транслятора. При связи процедуры, написанной на языке ассемблера, с языком высокого уровня, необходимо учитывать соглашение по передаче параметров.

Смешанные конвенции.

Существует конвенция передачи параметров STDCALL, отличающаяся и от C, и от PASCAL-конвенций, которая применяется для всех системных функций Win32 API. Здесь параметры помещаются в стек в обратном порядке, как в C, но процедуры должны очищать стек сами, как в PASCAL.

Еще одно отличие от C-конвенции – это быстрое или регистровое соглашение FASTCALL. В этом случае параметры в функции также передаются по возможности через регистры. Например, при вызове функции с шестью параметрами some_proc(a,b,c,d,e,f);

первые три параметра передаются соответственно в EAX, EDX, ECX, а только начиная с четвертого, параметры помещают в стек в обычном обратном порядке:

```
mov a, eax

mov b, edx

mov c, ecx

mov d, [ebp+8]

mov e, [ebp+12]

mov f, [ebp+16]
```

В случае если стек был задействован, освобождение его возлагается на вызываемую процедуру.

Выполнение работы.

Программа представляет собой код на c++ и внешнюю процедуру, реализованную на Ассемблере.

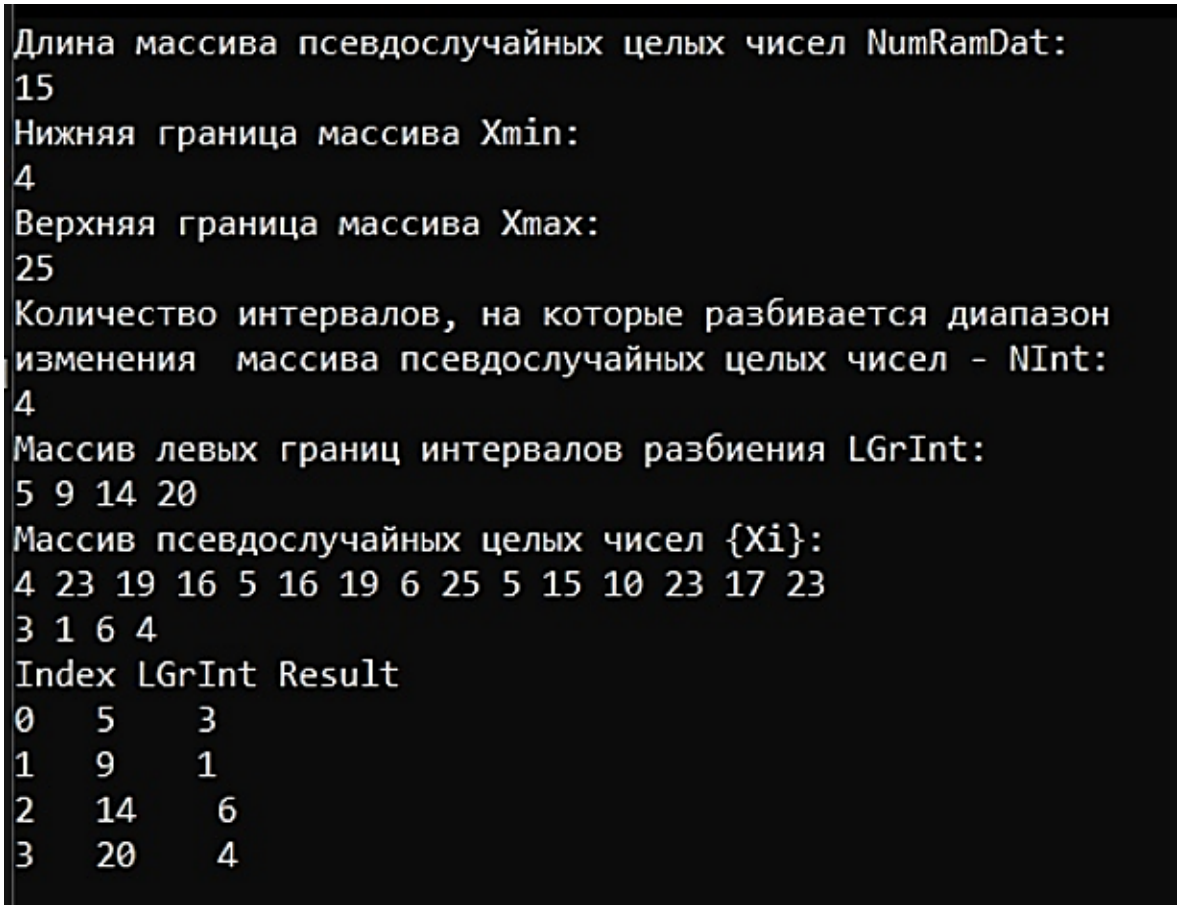
В main.cpp происходит ввод (значения считываются с клавиатуры до тех пор, пока не станут удовлетворять условиям – т.е. пока они не будут лежать в допустимых интервалах). В процессе считывания массив также упорядочивается, и уже отсортированный передаётся в процедуру lb6.asm, в которой происходит обработка этого массива псевдослучайных чисел.

Процедура представляет собой цикл (возвращается к метке cycle до тех пор, пока не будут обработаны все числа массива – с помощью инструкции loop). За определение границ значения ячейки массива отвечает метка toborder. В ней сравнивается число сначала с левой границей, затем (если оно больше границы) с последующими, до тех пор, пока оно не окажется меньше текущей границы. Таким образом, мы находим её ограничение сверху (текущая граница) и снизу (предыдущая граница).

В таком случае совершаем прыжок к метке quit, в которой увеличиваем значение найденного интервала в результирующем массиве.

Исходный код см. в приложении А.

Тестирование.



Вывод.

Изучен принцип связи программ на Ассемблере с кодом на языках программирования высокого уровня. Реализована программа, решающая поставленную задачу с помощью взаимосвязи языков программирования C++ и Ассемблера.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

```
#include <iostream>
#include <iomanip>
#include <random>
#include <fstream>
#include <clocale>
#include <algorithm>

extern "C" void func(int f_NumRamDat, int* f_X, int f_NInt, int
*f_LGrInt, int* result);

int main(){
    int NumRamDat, Xmax, Xmin, NInt, Dx;
    setlocale(LC_ALL, "");
    std::cout << "Длина массива псевдослучайных целых чисел NumRamDat:\n";
    std::cin >> NumRamDat;
    while (!(NumRamDat > 0 && NumRamDat < 16000)) {
        std::cout << "Длина массива псевдослучайных целых чисел должна быть <= 16K\n";
        std::cin >> NumRamDat;
    }
    int* X = new int[NumRamDat];
    std::cout << "Нижняя граница массива Xmin:"<<
std::endl;
    std::cin >> Xmin;
    std::cout << "Верхняя граница массива Xmax:"<<
std::endl;
    std::cin >> Xmax;
    while (Xmin >= Xmax) {
        std::cout << "Невозможный случай"<< std::endl;
        std::cin >> Xmax;
    }
    Dx = Xmax - Xmin;
    std::cout << "Количество интервалов, на которые разбивается диапазон\n"
          "изменения массива псевдослучайных целых чисел - NInt:"<< std::endl;
    std::cin >> NInt;
    while (NInt <= 0 || NInt > 24 || NInt >= Dx) {
        std::cout << "Недопустимое значение для NInt";
        std::cin >> NInt;
    }

    int* LGrInt = new int[NInt + 1];
    std::cout << "Массив левых границ интервалов разбиения LGrInt:"<< std::endl;
    for (int i = 0; i < NInt; i++) {
        std::cin >> LGrInt[i];
        while (LGrInt[i] > Xmax || LGrInt[i] <= Xmin) {
            std::cout << "Недопустимое значение для
LGrInt"<< std::endl;
```

```
            std::cin >> LGrInt[i];
        }
        while (i > 0 && LGrInt[i] < LGrInt[i - 1]) {
            std::swap(LGrInt[i--], LGrInt[i]);
        }
    }
    //LGrInt[NInt] = Xmax;

    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> distrib(Xmin, Xmax);

    std::cout << "Массив псевдослучайных целых чисел
{Xi}:"<< std::endl;
    for (int index = 0; index < NumRamDat; index++) {
        X[index] = distrib(gen);
        std::cout << X[index] << ' ';
    }
    std::cout << std::endl;

    auto result = new int[NumRamDat];
    for (int i = 0; i < NInt; i++) {
        result[i] = 0;
    }
    func( NumRamDat, X,  NInt, LGrInt, result);
    for (int i = 0; i < NInt; i++) {
        std::cout << result[i] << " ";
    }
    std::cout << std::endl;
    std::ofstream fout("ASM_output.txt");
    std::cout << "Index LGrInt Result" << std::endl;
    fout << "Index LGrInt Result" << std::endl;
    for (int i = 0; i < NInt; i++) {
        std::cout << i << "   " << LGrInt[i] << "    " << result[i] <<
std::endl;
        fout << i << "   " << LGrInt[i] << "    " << result[i] <<
std::endl;
    }
    fout.close();
    return 0;
}
```

Файл lb6.asm

```
.586
.MODEL FLAT, C
.CODE
func PROC C NumRamDat:dword, X:dword, NInt:dword, LGrInt:dword,
result:dword
push ESI
push EDI
push EAX
push EBX
push ECX

mov ESI, X
mov EDI, LGrInt
mov ECX, NumRamDat
mov EAX, 0
cycle:
     mov EBX, 0
toborder:
     cmp EBX, NInt
     jge quit
     push EAX
     mov EAX, [esi + 4 * eax]
     cmp EAX, [edi + 4 * ebx]
     pop EAX
     jl quit
     inc EBX
     jmp toborder
quit:
     dec EBX
     mov EDI, result
     push EAX
     mov EAX, [EDI + 4 * ebx]
     inc EAX
     mov [EDI + 4 * ebx], EAX
     pop EAX
     mov EDI, LGrInt
     inc EAX
```

```
loop cycle

pop ECX
pop EBX
pop EAX
pop EDI
pop ESI
ret
func ENDP
END
```