# ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

# «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»

**Кафедра МО ЭВМ**

# ОТЧЁТ

**по лабораторной работе № 6**

# по дисциплине «Организация ЭВМ и систем»

**Тема: Организация связи Ассемблера с ЯВУ на примере программы построения псевдослучайных целых чисел в заданные интервалы.**

Студент гр. 1303 Токун Г.С.

Преподаватель Ефремов М.А.

Санкт-Петербург 2022

# Цель работы.

Рассмотреть способ организации связи Ассемблера с ЯВУ. Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

# Задание.

Вариант 1.

На языке высокого уровня (Pascal или С) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND\_GEN (пpи его отсутствии программу датчика получить у пpеподавателя).

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.

* Длина массива псевдослучайных целых чисел — NumRanDat (*≤*

16*K, K* = 1024);

* Диапазон изменения массива псевдослучайных целых чисел

*Xmin, Xmax*, значения могут быть биполярные;

* Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел — NInt (*≤* 24);
* Массив левых границ интервалов разбиения LGrInt (должны

принадлежать итервалу [*Xmin, Xmax*]).

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.

# Выполнение работы.

Программа была написана с использованием двух ”модулей”: один написан на языке Си, в котором происходит считывание исходных данных и запись результатов, второй написан на языке Ассемблера — в нём происходит обработка данных. Для компиляции и линковки модулей была использована коллекция компиляторов GNU, команды объединены в Make- file.

В начале программы с помощью модуля на языке Си происходит считывание исходных данных, выделение памяти под массивы и их последующее заполнение согласно введённым данным; на каждом этапе проводится проверка данных — в случае некорректных данных программа выводит ошибку и завершается.

После вызывается процедура ассемблера process\_data, которая принимает исходные данные (результирующий массив, массив чисел, массив левых границ интервалов, количество чисел, количество левых границ). Данная процедура для каждого числа из массива вызывает процедуру find\_interval\_index, которая в свою очередь, начиная с последнего интервала, проверяет, входит ли данное число в текущий интервал, — номер интервала записывается в регистр rax (0 в случае отсутствия подходящего интервала). Далее (в случае нахождения интервала), ячейка результирующего массива необходимого интервала увеличивается на 1.

В конце концов, массив с результатом средствами языка Си записывается в файл ”results.txt”.

# Выводы

Рассмотрены способы организации связи Ассемблера с ЯВУ, получены практические навыки в написании программы, использующей язык Си и Ассемблера. Разработана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

# ПРИЛОЖЕНИЕ A КОД ПРОГРАММ

Название файла: main.c

#include <stdio.h> #include <stdlib.h> #include <time.h>

extern void process\_data(int \*result\_array, int \*source\_array,

int \*borders\_array, int number, int borders\_number);

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

int n = 0;

int x\_min = 0; int x\_max = 0;

int n\_int = 0;

printf("Enter: numbers' count, generation left border and right border, " "count of intervals\n");

scanf("%d %d %d %d", &n, &x\_min, &x\_max, &n\_int);

if (n = 0 | n > 16 \* 1024) { printf("Invalid numbers' count\n"); return 1;

} else if (x\_min = x\_max) {

printf("Invalid borders of number generation\n"); return 1;

} else if (n\_int = 0 | n\_int > 24) { printf("Invalid number of intervals\n"); return 1;

}

int \*n\_arr = malloc(n \* sizeof(int));

int \*int\_arr = malloc(n\_int \* sizeof(int));

printf("Enter left borders of intervals\n"); char c;

for (int i = 0; i < n\_int; +i) { scanf("%d%c", &int\_arr[i], &c);

if ((int\_arr[i] < x\_min | int\_arr[i] > x\_max) | (i > 0 & int\_arr[i] = int\_arr[i - 1])) {

printf("Invalid left border!\n"); goto error\_free\_sources;

}

}

int rand\_max = x\_max - x\_min + 1; for (int i = 0; i < n; +i) {

n\_arr[i] = x\_min + rand() % rand\_max;

}

int \*res\_arr = calloc(n\_int, sizeof(int)); process\_data(res\_arr, n\_arr, int\_arr, n, n\_int);

FILE \*f = fopen("results.txt", "w"); if (!f) {

printf("Cannot create file to write results"); goto error\_free\_result;

}

fputs("Generated numbers:\n", f); for (int i = 0; i < n; +i) {

fprintf(f, "%d ", n\_arr[i]);

}

fputs("\n\n", f);

fputs("Results:\n", f);

for (int i = 0; i < n\_int; +i) {

fprintf(f, "%d %d %d\n", i + 1, int\_arr[i], res\_arr[i]);

}

fclose(f); return 0;

error\_free\_result: free(res\_arr);

error\_free\_sources: free(n\_arr); free(int\_arr); return 1;

};

Название файла: lib.s

.global process\_data #; Input:

#; rdx > int \*borders\_array

#; rcx > int borders\_number #; rax > int number

#; Output:

#; rax > int interval\_index + 1 #; rax = 0 > not in interval find\_interval\_index:

find\_interval\_index\_loop: cmp eax, [rdx + rcx \* 4 - 4]

jge find\_interval\_index\_end loop find\_interval\_index\_loop

find\_interval\_index\_end:

mov rax, rcx ret

#; Input:

#; rdi > int \*result\_array (qword) #; rsi > int \*source\_array (qword) #; rdx > int \*borders\_array (qword) #; rcx > int count (dword)

#; r8 > int borders\_number (dword) process\_data:

push rax

process\_data\_loop:

lodsd

push rcx mov rcx, r8

call find\_interval\_index pop rcx

test rax, rax

jz continue\_loop

inc dword ptr [rdi + rax \* 4 - 4] continue\_loop:

loop process\_data\_loop

pop rax

ret

Название файла: Makefile

all: main

main: main.o lib.o

gcc main.o lib.o -o main -z noexecstack

main.o: main.c

gcc -c main.c

lib.o: lib.s

as lib.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o lib.o

clean:

rm -f \*.o main