**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №6**

**по дисциплине «Организация систем и ЭВМ»**

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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1303 |  | Сырцева Д.Д |
| Преподаватель |  | Ефремов М.А |

Санкт-Петербург

2022

## Цель работы.

Получить навыки программирования на языке Ассемблера. Разработать программу на ЯВУ с использованием языка Ассемблера, выполняющую построение частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

## Задание.

Вариант 23.

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 (пpи его отсутствии получить у пpеподавателя). 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.

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.

В ходе выполнения работы написаны два файла – main.c и lib.s – модуль, выполняющий обработку данных, написанный на языке Ассемблера.

Также был создан Makefile.

В файле main.c в функции main() происходит считывание данных, проверка на корректность введенных данных, передача данных в ассемблерный модуль для их обработки. Полученный результат выводится в файл(result.txt) средствами ЯВУ. Генерация чисел из заданного промежутка происходит с помощью функции rand().

В ассемблерном модуле обработка происходит следующим образом:

С помощью lodsd из массива сгенерированных чисел загружается двойное слово в регистр eax, далее, значение, хранящееся в eax сравнивается поочередно со значениями левых границ(начиная с последней), если это значение оказывается больше чем текущая левая граница, то происходит переход на метку end\_find(интервал найден), иначе же сравнение происходит со следующей левой границей. Когда все границы перебраны, то берется следующий символ и для него происходит проверка. Когда перебраны все символы, то происходит выход из ассемблерного модуля.

В метке end\_find значение, хранящееся в массиве res в конкретной ячейке увеличивается на 1, таким образом происходит подсчет. Измененный массив res теперь хранит результат выполнения работы.

## Тестирование.

## 

## Вывод.

В результате лабораторной работы была написана программа, корректно выполняющая формирование распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Рассмотрен способ организации связи Ассемблера с ЯВУ.

## ПРИЛОЖЕНИЕ А

## Название файла main.c.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

extern void func(int\* res, int\* nums, int k, int n, int\* arr);

int main() {

printf("Введите количество чисел\n");

int n;

scanf("%d", &n);

if (n <= 0 || n > 16 \* 1024) {

printf("Неверное количество чисел\n");

return 1;

}

printf("Введите диапазон генерации чисел\n");

int a, b;

scanf("%d %d", &a, &b);

if (a > b) {

printf("Неверный диапазон генерации чисел\n");

return 1;

}

printf("Введите количество интервалов разбиения\n");

int k;

scanf("%d", &k);

if (k <= 0) {

printf("Неверное количество интервалов разбиения\n");

return 1;

}

printf("Введите массив левых границ\n");

int\* arr = malloc(sizeof(int) \* k);

for (int i = 0; i < k; i++) {

scanf("%d", &arr[i]);

if (arr[i] < a || arr[i] > b || i > 0 && arr[i] < arr[i - 1]) {

printf("Некорректное значение левой границы\n");

free(arr);

return 1;

}

}

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

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

nums[i] = rand() % (b - a + 1) + a;

}

int\* res = calloc(k, sizeof(int));

func(res, nums, k, n, arr);

FILE\* f = fopen("result.txt", "w");

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

fprintf(f, "%d ", nums[i]);

}

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

for (int i = 0; i < k; i++) {

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

}

fclose(f);

free(arr);

free(nums);

free(res);

return 0;

}

## Название файла lib.s.

.global func

#RDI - int \*res, RSI - int \*nums, RDX - int k, RCX - int n,>func:

lodsd

push rcx

mov rcx, rdx

find\_interval:

cmp eax, [r8+rcx\*4-4]

jge end\_find

loop find\_interval

jmp exit

end\_find:

inc dword ptr [rdi+rcx\*4-4]

exit:

pop rcx

loop func

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 -o main.o

lib.o: lib.s

as lib.s -mmnemonic=intel -msyntax=intel -mnaked-re>

clean:

rm -rf \*.o main