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. 1383 _____ Депрейс А.С

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Рассмотреть способ организации связи Ассемблера с ЯВУ. Реализовать программу частотного распределения случайных чисел по заданным интервалам.

Задание.

Вариант 01.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND_GEN (при его отсутствии программу датчика получить у преподавателя).

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.

1. Длина массива псевдослучайных целыхчисел - NumRanDat (<= 16K, K=1024)

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] , значения могут быть биполярные;

3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)

4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]).

Выполнение работы

Программа поделена на 2 модуля.

Модуль на Си отвечает за считывание входных данных и их проверку, а также запись в консоль и файл результатов выполнения программы.

Модуль, написанный на языке Ассемблера реализует функцию, которая выполняет частотное распределение по заданным интервалам.Функция принимает(массив левых границ интервалов, массив случайно сгенерированных чисел, результирующий массив, количество случайных чисел и количество интервалов). Алгоритм работы: для каждого числа в массиве случайно сгенерированных чисел выполняется следующее: для каждой левой границы интервалов из числа вычитается левая граница, если расстояние > 0 и минимальное(на текущий момент) расстояние между числом и левой границей больше чем расстояние на текущей итерации, то запоминается индекс левой границы, а также перезписывается минимальное расстояние между числом и левой границей. Затем в результирующем массиве с индексом(левая граница с минимальным положительным расстоянием) прибавляется еденица.

После всех итераций со случайно сгенерированными числами результирующий массив будет заполнен.

Выводы

Рассмотрен способ организации связи Ассемблера с ЯВУ. Реализована программа частотного распределения случайных чисел по заданным интервалам.

Приложение А.

Исходный код программы.

Название файла: main.c

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

int* rand_nums_setup(int Xmin, int Xmax, int NumRandDat){
   int* nums_array = calloc(NumRandDat, sizeof(int));
   for (int i = 0; i < NumRandDat; i++)
   {
      nums_array[i] = Xmin + rand() % (Xmax - Xmin + 1);
   }
   return nums_array;
}

extern void asm_function(int* left_borders, int* rand_nums, int* result, int NumRandDat, int
Nint);

int main(){
   int NumRanDat = 0;
   int Xmin = 0;
   int Xmax = 0;
   int Nint = 0;
   int* left_borders;

   printf("Enter NumRanDat, Xmin, Xmax, Nint\n");
   scanf("%d %d %d %d", &NumRanDat, &Xmin, &Xmax, &Nint);

   if(NumRanDat < 0 || NumRanDat > (1024*16)){
      printf("Wrong number of random numbers\n");
      return 0;
   }

   if(Nint < 0 || Nint > 24){
      printf("Wrong number of intervals\n");
```

```
    return 0;
}

if(Xmin > Xmax){
    printf("Wrong range of random numbers\n");
    return 0;
}

left_borders = (int*)calloc(Nint, sizeof(int));
printf("Enter left borders\n");
for (int i = 0; i < Nint; i++)
{
    scanf("%d", left_borders + i);
    if (left_borders[i] < Xmin || left_borders[i] > Xmax)
    {
        printf("Wrong interval boundary\n");
        free(left_borders);
        return 0;
    }

}


int* rand_nums = rand_nums_setup(Xmin,Xmax,NumRanDat);
printf("Array of random numbers\n[");
for (int i = 0; i < NumRanDat - 1; i++)
{
    printf("%d, ", rand_nums[i]);
}
printf("%d]\n", rand_nums[NumRanDat - 1]);

int* result = calloc(Nint, sizeof(int));

asm_function(left_borders, rand_nums, result, NumRanDat, Nint);
```

```
    for (int i = 0; i < Nint; i++)
    {
        printf("Interval num) %d    Left border) %d    Nums in interval) %d\n",i + 1, left_borders[i], result[i]);
    }

    FILE *file = fopen("result.txt","w");
    for (int i = 0; i < Nint; i++)
    {
        fprintf(file, "Interval num) %d    Left border) %d    Nums in interval) %d\n",i + 1, left_borders[i], result[i]);
    }
    fclose(file);

    free(result);
    free(left_borders);
    free(rand_nums);
}
```

Название файла: asm_module.s

```
.global asm_function

# rdi <-- left_borders_arr
# rsi <-- rand_nums_arr
# rdx <-- result_arr
# rcx <-- NumRandDat
# r8 <-- Nint

asm_function:
    push rax

sorting_out_numbers:
    lodsd
```

```
    push rcx

    mov rcx, r8
    jmp finding_interval
interval_found:
    pop rcx

    cmp r10, 0
    jl next_number

    incq [rdx][r10 * 4 - 4]

next_number:
    loop sorting_out_numbers

    pop rax
    ret


# eax <-- random_number
# r10 --> index of interval + 1
finding_interval:
    mov r10, -1 # interval index
    mov r11d, -1 # distance to interval
    mov r12, 0  # first flag
sorting_out_intervals:

    mov ebx, eax
    sub ebx, [rdi][rcx * 4 - 4]
    cmp ebx, 0
    jl next_iteration

if_first_iter:
    cmpq r12, 0
```

```
    jne not_first_iter
    movq r10, rcx
    mov r11d, ebx
    movq r12, 1
    jmp next_iteration
not_first_iter:

    cmp r11d, ebx
    jle next_iteration

    movq r10, rcx
    mov r11d, ebx



next_iteration:
    loop sorting_out_intervals

    jmp interval_found
```