

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

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.

Студентка гр. 1303

Андреева Е.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Получить навыки программирования на языке Ассемблера. Разработать программу на ЯВУ с использованием языка Ассемблера.

Задание.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

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 целых чисел - NumRanDat ($\leq 16K$, $K=1024$)
2. Диапазон изменения массива псевдослучайных целых чисел $[X_{\min}, X_{\max}]$, 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 $[X_{\min}, X_{\max}]$).

Результаты:

1. Текстовый файл, строка которого содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк равно числу интервалов разбиения.

2. График, отражающий распределение чисел по интервалам.

(необязательный результат)

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 главную программу, написанную на ЯВУ.

Ход работы.

На языке c++ реализовано считывание начальных данных. Левые границы находятся в массиве `int_arr`, а генерируемые числа добавляются в массив `n_arr`. Также для хранения результата был создан массив `res_arr`.

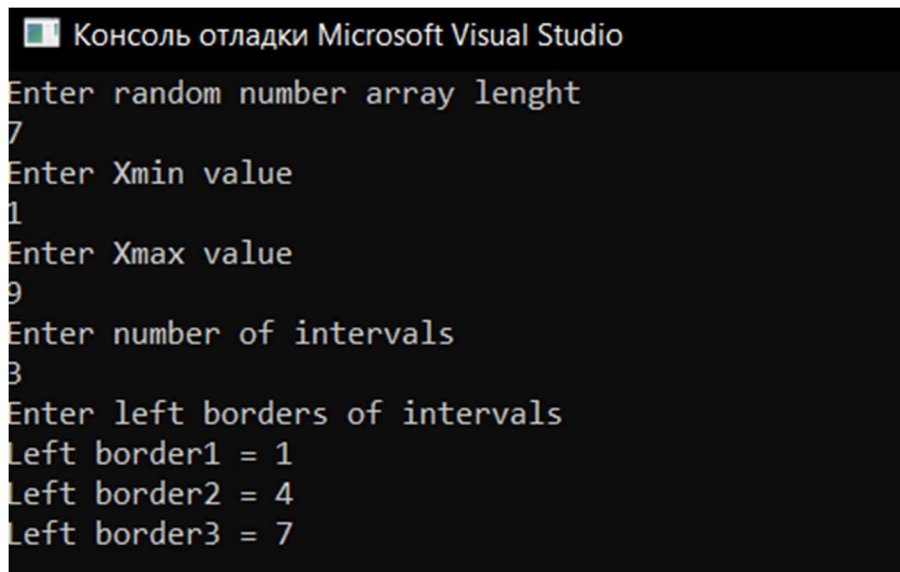
В ассемблерный модуль в процедуру `function` передаются указатель на массив чисел `array`, его длина `len`, указатель на массив левых границ `LGrInt`, его размер `NInt`, указатель на результирующий массив `result`. В процедуре для каждого элемента массива `array` находится интервал, в который попадает этот элемент и результат записывается в массив `result`.

После выполнения процедуры результат выводится в файл `out.txt`.

Исходный код программы см. в приложении А.

Тестирование.

Работа программы с заданными условиями представлена на рис. 1



```
Консоль отладки Microsoft Visual Studio
Enter random number array lenght
7
Enter Xmin value
1
Enter Xmax value
9
Enter number of intervals
3
Enter left borders of intervals
Left border1 = 1
Left border2 = 4
Left border3 = 7

Generated numbers:
5 5 5 9 2 9 7

Results:
1. left border:1  numbers - 1
2. left border:4  numbers - 3
3. left border:7  numbers - 3
```

Рис.1 – Результат работы программы

Вывод.

В ходе выполнения лабораторной работы были получены навыки программирования на языке Ассемблера. Была разработана программа на ЯВУ с использованием языка Ассемблера.

Приложение А

Исходный код программы

Название файла: main.cpp

```
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <fstream>

std::ofstream file("out.txt");

extern "C" {void function(int* Array, int len, int* LGrInt, int
NInt, int* answer); }

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));

    int NumRanDat = 0;
    int x_min = 0;
    int x_max = 0;

    int NInt = 0;

    std::cout << "Enter random number array lenght\n";
    std::cin >> NumRanDat;
    if (NumRanDat <= 0 || NumRanDat > 16 * 1024) {
        std::cout << "Invalid random number array lenght\n";
        return 1;
    }

    std::cout << "Enter Xmin value\n";
    std::cin >> x_min;
    std::cout << "Enter Xmax value\n";
    std::cin >> x_max;
    if (x_min >= x_max) {
        std::cout << "Invalid Xmin and Xmax values\n";
        return 1;
    }

    std::cout << "Enter number of intervals\n";
    std::cin >> NInt;
    if (NInt <= 0 || NInt > 24) {
        std::cout << "Invalid number of intervals\n";
        return 1;
    }

    int *n_arr = new int[NumRanDat];
    int *int_arr = new int[NInt];

    std::cout << "Enter left borders of intervals\n";
```

```

for (int i = 0; i < NInt; ++i) {
    std::cout << "Left border" << i + 1 << " = ";
    std::cin >> int_arr[i];
    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");
        delete[] n_arr;
        delete[] int_arr;
        return 1;
    }
}

int rand_max = x_max - x_min + 1;
for (int i = 0; i < NumRandat; ++i) {
    n_arr[i] = x_min + rand() % rand_max;
}

int *res_arr = new int[NInt] {0};
function(n_arr, NumRandat, int_arr, NInt, res_arr);

file << "Generated numbers:\n";
for (int i = 0; i < NumRandat; ++i) {
    file << n_arr[i] << " ";
}
file << "\n\nResults:\n";
for (int i = 0; i < NInt; ++i) {
    file << i + 1 << ". left border:" << int_arr[i] << " numbers
- " << res_arr[i] << "\n";
}

file.close();
delete[] n_arr;
delete[] int_arr;
delete[] res_arr;

return 0;
};

```

Название файла: task.asm

.586p

.MODEL FLAT, C

.CODE

function PROC C USES EDI ESI, array:dword, len:dword, LGrInt:dword,
NInt:dword, result:dword

```

push eax
push ebx
push ecx
push edi
push esi

```

```

mov ecx, len
mov esi, array
mov edi, LGrInt
mov eax, 0

lp:
mov ebx, 0 ; количество пройденных интервалов
iter:
    cmp ebx, NInt
    jge out_iter ; если количество пройденных интервалов больше, чем
NInt, то выходим в out_iter

    push eax
    mov eax, [esi + 4 * eax] ; в eax элемент массива array
    cmp eax, [edi + 4 * ebx] ; сравниваем элемент массива array с
левой границей
    pop eax
    jl out_iter ; если меньше, то выходим в out_iter
    inc ebx ; если больше, то переходим на следующий интервал
    jmp iter

out_iter:
    dec ebx ; уменьшаем номер интервала на 1

    cmp ebx, -1
    je to_next_num
    mov edi, result
    push eax
    mov eax, [edi + 4 * ebx] ; в eax помещаем элемент массива result
с номером ebx
    inc eax ; увеличиваем этот элемент на 1
    mov [edi + 4 * ebx], eax
    pop eax
    mov edi, LGrInt

to_next_num:
    inc eax

```

```
loop lp
```

```
pop esi
```

```
pop edi
```

```
pop ecx
```

```
pop ebx
```

```
pop eax
```

```
ret
```

```
function ENDP
```

```
END
```



```
0000          AStack SEGMENT STACK
0000 0400[          DB 1024 DUP(?)
    ??
    ]

0400          AStack ENDS


0000          DATA SEGMENT
0000 0000          KEEP_CS DW 0
0002 0000          KEEP_IP DW 0
0004 48 65 6C 6C 6F 21          MESSAGE DB 'Hello!', 0dh, 0ah, '$'
    0D 0A 24
000D 65 6E 64 0D 0A 24          END_MES DB 'end', 0dh, 0ah, '$'
0013 20 24          EMPTY DB ' ', '$'
0015          DATA ENDS


0000          CODE SEGMENT
          ASSUME CS:CODE, DS:DATA, SS:AStack


0000          WriteMsg PROC NEAR
0000 B4 09          mov AH, 9
0002 CD 21          int 21h
0004 C3          ret
0005          WriteMsg ENDP


0005          FUNC PROC FAR
```

0005 50	push ax
0006 53	push bx
0007 51	push cx
0008 52	push dx
0009 1E	push ds
000A BA 0004 R	mov dx, OFFSET MESSAGE
000D B9 0004	mov cx, 4
0010	lp:
0010 E8 0000 R	call WriteMsg
0013 E2 FB	loop lp
0015 B0 00	mov al, 0
0017 B4 86	mov ah, 86h
0019 33 C9	xor cx, cx
001B BA 2710	mov dx, 10000
001E CD 15	int 15h
0020 BA 000D R	mov dx, OFFSET END_MES
0023 E8 0000 R	call WriteMsg
0026 1F	pop ds
0027 5A	pop dx
0028 59	pop cx
0029 5B	pop bx
002A 58	pop ax
002B B0 20	mov al, 20h
002D E6 20	out 20h, al

```
002F CF                                ired
0030                                FUNC ENDP

0030                                MAIN PROC FAR
0030 1E                                push ds
0031 2B C0                            sub ax, ax
0033 50                                push ax
0034 B8 ---- R                        mov ax, DATA
0037 8E D8                            mov ds, ax

0039 B4 35                            mov ah, 35h
003B B0 08                            mov al, 08h
003D CD 21                            int 21h
003F 89 1E 0002 R                      mov KEEP_IP, bx
0043 8C 06 0000 R                      mov KEEP_CS, es

0047 1E                                push ds
0048 BA 0005 R                        mov dx, OFFSET FUNC
004B B8 ---- R                        mov ax, SEG FUNC
004E 8E D8                            mov ds, ax
0050 B4 25                            mov ah, 25h
0052 B0 08                            mov al, 08h
0054 CD 21                            int 21h
0056 1F                                pop ds

0057 CD 08                            int 08h
```

0059	FA	cli
005A	1E	push ds
005B	8B 16 0002 R	mov dx, KEEP_IP
005F	A1 0000 R	mov ax, KEEP_CS
0062	8E D8	mov ds, ax
0064	B4 25	mov ah, 25h
0066	B0 08	mov al, 08h
0068	CD 21	int 21h
006A	1F	pop ds
006B	FB	sti
006C	B4 4C	mov ah, 4ch
006E	CD 21	int 21h
0070		MAIN ENDP
0070		CODE ENDS
		END MAIN

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0400	PARA		STACK
CODE	0070	PARA		NONE
DATA	0015	PARA		NONE

Symbols:

N a m e	Type	Value	Attr
EMPTY	L BYTE	0013	DATA
END_MES	L BYTE	000D	DATA
FUNC	F PROC	0005	CODE Length = 002B
KEEP_CS	L WORD	0000	DATA
KEEP_IP	L WORD	0002	DATA
LP	L NEAR	0010	CODE
MAIN	F PROC	0030	CODE Length = 0040
MESSAGE	L BYTE	0004	DATA

WRITEMSG N PROC 0000 CODE Length =
0005

@CPU TEXT 0101h
@FILENAME TEXT lab5
@VERSION TEXT 510

92 Source Lines

92 Total Lines

17 Symbols

48016 + 461291 Bytes symbol space free

0 Warning Errors

0 Severe Errors