**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Операционные системы»**

**Тема: Исследование структур загрузочных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Ларин А. |
| Преподаватель |  | Ефремов М.А, |

Санкт-Петербург

2020

**Цель работы.**

Исследование различий в структурах исходных текстов модулей типов **.COM** и **.EXE**, структур файлов загрузочных модулей и способов их загрузки в основную память.

**Выполнение**

Написан код .COM модуля, который определяет тип PC и версию системы. В нем читаются байты из ROM BIOS, и интерпретируются в соответствие с таблицей. Затем при помощи системного прерывания запрашивается информация о системе и так же выводится на экран. Данный код был собран в .COM модуль, а так же в «плохой» .EXE модуль в результате чего было получено предупреждение линковщика: «LINK : warning L4021: no stack segment».

Результаты исполнения

|  |  |  |
| --- | --- | --- |
| .COM | «плохой» .EXE | «хороший» .EXE |
|  |  |  |

Код представлен в приложении А.

**Отличия исходных текстов COM и EXE программ**

1. COM содержит ровно один сегмент
2. EXE содержит произвольное количество сегментов
3. ORG 100h, для размещения PSP
4. Для COM некорректно указание адреса сегмента

Шестнадцатеричный xxd вид скомпонованных модулей представлен в приложении Б

**Отличие форматов файлов COM и EXE модулей**

1. COM файл содержит единственный сегмент с кодом и данными. В памяти код располагается начиная с 100h
2. Плохой EXE файл содержит в себе только один сегмент с кодом и данными, и тот начинается с адреса 300h. С адреса 0h распологается метаинформация и множество пустых ячеек памяти
3. Плохой EXE содержит только один сегмент. Так же информация начале файла разнится и плохой EXE занимает больше памяти

**Загрузка COM модуля в основную память**

**Тестирование.**

**1.**

Enter amount of elements:

5

Enter amount of intervals:

2

Enter lower border:

-5

Enter upper border:

5

Enter 1 left borders:

1

NInt lGrInt Num

0 -5 4

1 1 1

**2.**

Enter amount of elements:

10000

Enter amount of intervals:

10

Enter lower border:

-10

Enter upper border:

10

Enter 9 left borders:

-8 -6 -4 -2 0 2 4 6 8

NInt lGrInt Num

0 -10 2247

1 -8 656

2 -6 740

3 -4 776

4 -2 791

5 0 771

6 2 753

7 4 690

8 6 589

9 8 1987

**3.**

Enter amount of elements:

10000

Enter amount of intervals:

10

Enter lower border:

-50

Enter upper border:

-30

Enter 9 left borders:

-48 -46 -44 -42 -40 -38 -36 -34 -32

NInt lGrInt Num

0 -50 2247

1 -48 656

2 -46 740

3 -44 776

4 -42 791

5 -40 771

6 -38 753

7 -36 690

8 -34 589

9 -32 1987

**Выводы.**

В результате работы были разобраны некоторые концепции совмещения языка ассемблера с ЯВУ. Были изучены Способу передачи аргументов в функцию и возвращение значений. Была написана рабочая программа для исследования распределения псевдослучайных величин.

Приложение

EVM\_6.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include<iostream>

#include"RANDOM.H"

extern "C"

{

void MAS\_FUNC(int n, int ni, int xmi, int xma, int\* nums, int\* lB, int\* res);

}

//using namespace std;

int main()

{

dnk\_randomize();

int n = 0;

int ni = 0;

int xMin = 0;

int xMax = 0;

std::cout << "Enter amount of elements:\n";

std::cin >> n;

if (n > 16 \* 1024) {

std::cout << "amount of elements shold be no bigger than " << 16 \* 1024; exit(1);

}

std::cout << "Enter amount of intervals:\n";

std::cin >> ni;

if (ni > 24) {

std::cout << "amount of intervals shold be no bigger than " << 24; exit(1);

}

std::cout << "Enter lower border:\n";

std::cin >> xMin;

std::cout << "Enter upper border:\n";

std::cin >> xMax;

int \*nums = new int[n];

int \*lB = new int[ni];

int \*res = new int[ni];

for (int i = 0; i < ni; i++)res[i] = 0;

std::cout << "Enter "<<ni-1<<" left borders:\n";

for (int i = 0; i < ni-1; i++) {

//check for inside borders

std::cin >> lB[i];

if (lB[i] > xMax || lB[i] < xMin) {

std::cout << "Left borders shoult be within min-max interval"; exit(1);

}

}

lB[ni - 1] = xMax;

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

nums[i] = round(dnk\_normal((double)((xMin+xMax)/2), (double)((xMax-xMin)/2)));

// std::cout << nums[i] << " ";

if (nums[i] > xMax)nums[i] = xMax-1;

if (nums[i] < xMin)nums[i] = xMin+1;

}

std::cout << std::endl;

MAS\_FUNC(n, ni, xMin, xMax,nums, lB,res);

printf("NInt\tlGrInt\tNum\n");

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

printf("%d\t%d\t%d\n", i, i ? lB[i - 1] : xMin, res[i]);

}

}

Inc.asm

.686

.MODEL FLAT, C

.STACK

.DATA

;-----------Local data------------------------------

.CODE

;-----------External usage--------------------------

;-----------Function definitions--------------------

MAS\_FUNC PROC C n:dword, ni:dword, xmi:dword, xma:dword,nums:dword, bordArr:dword, inclArr:dword

mov ecx,0

mov ebx,[nums]

mov esi,[bordArr]

mov edi,[inclArr]

lp:

nop

mov eax,[ebx]

push ebx;

mov ebx,0;edge num

lp1:

mov edx,ebx;

shl edx,2;

;mov ebx,[esi+edx]

cmp eax,[esi+edx];more then next border

jg ink

jmp ent

ink:

inc ebx

jmp lp1

ent:

add edx,edi

mov eax,[edx]

inc eax

mov [edx],eax;element lies in edx section

pop ebx;

add ebx,4

inc ecx

cmp ecx,n

jl lp

ret

MAS\_FUNC ENDP

END