**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №8  
дисциплина: Архитектура вычислительных систем  
тема: «Способы вызова ассемблерных подпрограмм в языках высокого уровня»

Выполнил: ст. группы ВТ-31  
Ковалёв И. Д.  
Проверил: Осипов О. В.

**Белгород 2019**

**Цель работы:** изучение способов вызова подпрограмм, написанных на разных языках

программирования посредством dll-библиотек.

**Задания для выполнения к работе**

1. Написать и отладить подпрограммы на masm32 в разных стилях вызова для решения задачи соответствующего варианта. Глобальные переменные в подпрограммах использовать не разрешается. Если нужна дополнительная память, выделять её в стеке.
2. Подпрограммы собрать и скомпилировать в виде dll-библиотеки.
3. Библиотека должна содержать:

подпрограммы в стилях stdcall, cdecl, fastcall, написанные на ассемблере без явного перечисления аргументов в заголовке;

Подпрограммы в стилях stdcall, cdecl, написанные, наоборот, с перечислением аргументов в заголовке подпрограммы. 3 Подключить все подпрограммы из dll-библиотеки к проектам на C# и С++ статическим и динамическим способом. Убедиться в правильности вызова всех подпрограмм.

1. Написать подпрограмму для решения задачи варианта с использованием ассемблерной вставки на языке C++.
2. Написать подпрограммы для решения задачи варианта с использованием обычного высокоуровнего языка C# и C++ (или любого другого).
3. Сравнить скорость выполнения полученных подпрограмм на одних и тех же тестовых данных. Для сравнения выбрать: подпрограмму на ассемблере в masm32 (какую-нибудь одну из пяти), вызываемую из программы на языке C++ или C#; подпрограмму на C#; подпрограмму на C++; подпрограмму на С++ с использованием ассемблерной вставки. Построить на одной плоскости графики зависимости времени выполнения подпрограмм от длины массивов (не менее 10 точек для каждой подпрограммы). Для замера лучше передавать в подпрограммы массивы большой длины. Время замерять в миллисекундах с помощью API-функции GetTickCount(). Проверить, что подпрограммы при одинаковых тестовых данных выдают одинаковый результат. Для заполнения массивов использовать генератор случайных чисел.

**Задание:**

Сортировка части массива чисел с индексами от start до end включительно.

Отсортированный массив (элементы start…end) записать в res и возвратить

его адрес. Исходный массив a оставить без изменений. Под массив res

зарезервировать память в необходимом размере, но не больше, чем нужно.

Пример: a = {4, 5, 4, 2, 5, 7, 5, 6, 5, 3, 5, 6}, start = 4, end = 8;

res = {5, 5, 5, 6, 7} (сортировка по не убыванию). Длина массива res равна 5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |  |  |  |  |  |  |  |
|  | Кол-во элементов | | 1000 | 2000 | | 4000 | 8000 | 16000 | 32000 | 64000 | 128000 | 256000 | 512000 |
|  | C, сек | | 0 | 0 | | 0,093 | 0,313 | 1,234 | 4,891 | 19,515 | 78,156 | 312,672 | 1250,688 |
|  | dll, сек | | 0 | 0,016 | | 0,016 | 0,078 | 0,281 | 1,125 | 4,485 | 17,937 | 71,734 | 286,936 |
|  | \_\_asm{} , сек | | 0 | 0 | | 0,016 | 0,0858 | 0,3091 | 1,2375 | 4,9335 | 19,7307 | 78,9074 | 315,6296 |
|  | |  |  |  |  | |  |  |  |  |  |  |  |
|  | |  |  |  |  | |  |  |  |  |  |  |  |
|  | |  | | | | | | | | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |

Содержимое dll-библиотеки:

.386

.model flat, stdcall

option casemap :none

.code

DLLMain proc hlnstDLL:DWORD, reason:DWORD, unused:DWORD

mov EAX, 1

ret

DLLMain endp

LibMain proc hlnstDLL:DWORD, reason:DWORD, unused:DWORD

mov EAX, 1

ret

LibMain endp

swap proc

push eax

push ebx

push ecx

push edx

push esi

push ebp

mov ebp, [esp + 36]

mov esi, [esp + 32]

mov ebx, [esp + 28]

mov eax, [ebp + 4 \* esi]

mov edx, [ebp + 4 \* ebx]

mov ecx,edx

mov edx,eax

mov eax,ecx

mov [ebp + 4 \* esi], eax

mov [ebp + 4 \* ebx], edx

pop ebp

pop esi

pop edx

pop ecx

pop ebx

pop eax

ret 12

swap endp

sort proc

push eax

push ebx

push ecx

push edx

push esi

push ebp

mov ecx, [esp + 28]

xor esi, esi

xor edx, edx

xor ebp, ebp

mov esi, -1

i\_cycle:

add esi, 1

mov ebp, esi

mov edx, esi

j\_cycle:

add edx, 1

mov eax, [ecx + edx \* 4]

mov ebx, [ecx + ebp \* 4]

cmp eax, ebx

jnl not\_if1

mov ebp, edx

not\_if1:

mov eax, [esp + 32]

sub eax, 1

cmp edx, eax

jl j\_cycle

cmp ebp, esi

je not\_if2

push ecx

push esi

push ebp

call swap

mov ebp, esi

not\_if2:

mov eax, [esp+32]

sub eax, 2

cmp esi, eax

jl i\_cycle

pop ebp

pop esi

pop edx

pop ecx

pop ebx

pop eax

ret 8

sort endp

copy proc

push ebp

mov ebp, esp

add ebp, 4

push esi

push edi

push ecx

push eax

push edx

mov esi, [ebp + 4 + 12]

mov eax, [ebp + 4]

mov ecx, 4

mul ecx

add esi, eax

mov edi, [ebp + 4 + 8]

mov ecx, [ebp + 4 + 4]

sub ecx, [ebp + 4]

inc ecx

cld

rep movsd

pop edx

pop eax

pop ecx

pop edi

pop esi

pop ebp

ret 16

copy endp

sort\_stdcall proc

push ebp ; пролог

mov ebp, esp ; сохранение адреса вершины стека в регистре ebp - пролог

push [ebp + 8 + 12] ; \*a

push [ebp + 8 + 8] ; \*res

push [ebp + 8 + 4] ; fin

push [ebp + 8] ; strt

call copy

mov eax, [ebp + 8 + 4]

sub eax, [ebp + 8]

inc eax ; eax = size

push eax ; size

push [ebp + 8 + 8] ; адрес массива res

call sort

mov eax, [ebp + 8 + 8] ; возврат адреса массива res

leave ; эпилог

ret 16

sort\_stdcall endp

sort\_cdecl proc

push ebp ; пролог

mov ebp, esp ; сохранение адреса вершины стека в регистре ebp - пролог

push [ebp + 8 + 12] ; \*a

push [ebp + 8 + 8] ; \*res

push [ebp + 8 + 4] ; end

push [ebp + 8] ; start

call copy

mov eax, [ebp + 8 + 4]

sub eax, [ebp + 8]

inc eax ; eax = size

push eax ; size

push [ebp + 8 + 8] ; адрес массива res

call sort

mov eax, [ebp + 8 + 8] ; возврат адреса массива res

leave ; эпилог

ret

sort\_cdecl endp

sort\_fastcall proc

push ebp ; пролог

mov ebp, esp ; сохранение адреса вершины стека в регистре ebp - пролог

push [ebp + 8 + 4] ; \*a

push [ebp + 8] ; \*res

push edx ; end

push ecx ; start

call copy

sub edx, ecx ; edx = size

inc edx

push edx ; size

push [ebp + 8] ; адрес массива res

call sort

mov eax, [ebp + 8] ; возврат адреса массива res

leave ; эпилог

ret 8

sort\_fastcall endp

sort\_stdcall\_args proc strt:dword, fin:dword, res:dword, a:dword

push a

push res

push fin

push strt

call copy

mov eax, fin

sub eax, strt

inc eax ; eax = size

push eax ; size

push res ; адрес массива res

call sort

mov eax, res ; возврат адреса массива res

ret 16

sort\_stdcall\_args endp

sort\_cdecl\_args proc c strt:dword, fin:dword, res:dword, a:dword

push a

push res

push fin

push strt

call copy

mov eax, fin

sub eax, strt ; eax = size

inc eax

push eax ; size

push res ; адрес массива res

call sort

mov eax, res ; возврат адреса массива res

ret

sort\_cdecl\_args endp

end LibMain

Демонстрационная программа:

#include <iostream>

#include <time.h>

#include <Windows.h>

#pragma comment(lib, "lab8.lib")

/\*

\_sort\_stdcall@16 = \_sort\_stdcall@0

sort\_cdecl

@sort\_fastcall@16 = \_sort\_fastcall@0

\_sort\_stdcall\_args@16

sort\_cdecl\_args

\*/

extern "C" \_\_declspec(dllimport) int\* \_stdcall sort\_stdcall(int start, int end, int\* res, int\* a);

extern "C" \_\_declspec(dllimport) int\* \_cdecl sort\_cdecl(int start, int end, int\* res, int\* a);

extern "C" \_\_declspec(dllimport) int\* \_fastcall sort\_fastcall(int start, int end, int\* res, int\* a);

extern "C" \_\_declspec(dllimport) int\* \_stdcall sort\_stdcall\_args(int start, int end, int\* res, int\* a);

extern "C" \_\_declspec(dllimport) int\* \_cdecl sort\_cdecl\_args(int start, int end, int\* res, int\* a);

using namespace std;

int\* generateRandomArray(int \* array, int length) {

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

array[i] = rand() % 100;

array[i] -= 50;

}

return array;

}

void printArray(int\* array, int length) {

for (int i = 0; i < length; i++)

cout << array[i] << " ";

}

void initArray(int\* array, int length) {

for (int i = 0; i < length; i++)

array[i] = 0;

}

int\* sort(int \* a, int length) {

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

int min = i;

for (int j = i + 1; j <= length; j++) {

if (a[j] < a[min]) {

min = j;

}

}

int t = a[i];

a[i] = a[min];

a[min] = t;

}

return a;

}

int\* asmSort(int strt, int fin, int\* res, int\* a) {

\_asm {

pushad

mov esi, a

mov eax, strt

mov ecx, 4

mul ecx

add esi, eax

mov edi, res

mov ecx, fin

sub ecx, strt

inc ecx

cld

rep movsd

mov esi, -1

i\_cycle:

add esi, 1

mov ebp, esi

mov edx, esi

j\_cycle :

add edx, 1

mov eax, [ecx + edx \* 4]

mov ebx, [ecx + ebp \* 4]

cmp eax, ebx

jnl not\_if1

mov ebp, edx

not\_if1 :

mov eax, [esp + 32]

sub eax, 1

cmp edx, eax

jl j\_cycle

cmp ebp, esi

je not\_if2

push ecx

push esi

push ebp

mov ebp, esi

not\_if2 :

mov eax, [esp + 32]

sub eax, 2

cmp esi, eax

jl i\_cycle

popad

}

return res;

}

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

int a[10], b[8];

generateRandomArray(a, 10);

cout << "Проверка работоспособности статически подключенных функций:" << endl;

int\* result;

initArray(b, 8);

cout << "stdcall без перечисления агрументов:" << endl;

generateRandomArray(a, 10);

result = sort\_stdcall(1, 8, b, a);

printArray(result, 8); cout << endl;

cout << "cdecl без перечисления агрументов:" << endl;

generateRandomArray(a, 10);

sort\_cdecl\_args(1, 8, b, a);

printArray(result, 8); cout << endl;

cout << "fastcall без перечисления агрументов:" << endl;

generateRandomArray(a, 10);

sort\_fastcall(1, 8, b, a);

printArray(result, 8); cout << endl;

cout << "stdcall перечислением агрументов:" << endl;

generateRandomArray(a, 10);

result = sort\_stdcall\_args(1, 8, b, a);

printArray(result, 8); cout << endl;

cout << "fastcall с перечислением агрументов:" << endl;

generateRandomArray(a, 10);

sort\_cdecl\_args(1, 8, b, a);

printArray(result, 8); cout << endl;

typedef int\* (\_stdcall \* \_sort\_stdcall) (int start, int end, int\* res, int\* a);

typedef int\* (\_cdecl \* \_sort\_cdecl) (int start, int end, int\* res, int\* a);

typedef int\* (\_fastcall\* \_sort\_fastcall) (int start, int end, int\* res, int\* a);

char dllName[] = "lab8.dll";

HMODULE h = LoadLibraryA(dllName);

cout << endl << "Проверка работоспособности динамически подключенных функций:" << endl;

cout << "stdcall:" << endl;

\_sort\_stdcall sort\_stdcall = (\_sort\_stdcall)GetProcAddress(h, "\_sort\_stdcall@16");

generateRandomArray(a, 10);

result = sort\_stdcall(1, 8, b, a);

printArray(result, 8);

cout << endl << "cdecl:" << endl;

\_sort\_cdecl sort\_cdecl = (\_sort\_cdecl)GetProcAddress(h, "sort\_cdecl");

generateRandomArray(a, 10);

result = sort\_cdecl(1, 8, b, a);

printArray(result, 8);

cout << endl << "fastcall" << endl;

\_sort\_fastcall sort\_fastcall = (\_sort\_fastcall)GetProcAddress(h, "@sort\_fastcall@16");

generateRandomArray(a, 10);

result = sort\_fastcall(1, 8, b, a);

printArray(result, 8);

cout << endl;

cout << "Сравним время выполнения сортировок:" << endl;

for (int size = 1000; size < 1000000; size \*= 2) {

int\* a = new int[size];

int\* b = new int[size];

generateRandomArray(a, size);

cout << "Количество элементов в массиве: " << size << endl;

DWORD t = GetTickCount();

sort(a, size - 1);

t = GetTickCount() - t;

cout << "Время сортировки функцией на Си: " << (float)t / 1000 << " с." << endl;

t = GetTickCount();

sort\_stdcall(0, size - 1, b, a);

t = GetTickCount() - t;

cout << "Время сортировки функцией из dll: " << (float)t / 1000 << " с." << endl;

t = GetTickCount();

asmSort(0, size - 1, b, a);

t = GetTickCount() - t;

cout << "Время сортировки ассемблерной вставки: " << (float)t / 1000 << " с." << endl;

}

}

Результаты работы программы:

