**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №7**

**Дисциплина: Способы вызова ассемблерных подпрограмм**

**в языках высокого уровня**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

**Белгород 2019**

**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. 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.
4. Написать подпрограмму для решения задачи варианта с использованием ассемблерной вставки на языке C++.
5. Написать подпрограммы для решения задачи варианта с использованием обычного высокоуровнего языка C# и C++ (или любого другого).
6. 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.

1. В отчёт включить весь исходный код и графики.
2. Сделать выводы по работе.

**Формулировка задания**

|  |
| --- |
| Пузырьковая сортировка по не убыванию.  int\* sort(int start, int end, int\* res, int\* a). |

1.

sort proc a:dword, \_start:dword, \_end:dword, res:dword

local Hheap:dword, i:dword, indexOfLastUnsortedElement:dword, swapped:dword

pusha

finit

mov ecx, \_end

sub ecx, \_start

mov i, ecx

fild i

mov i, 4

fimul i

fistp i

mov ecx, i

add ecx, 100

invoke HeapCreate, 0, ecx, ecx

mov Hheap, eax

mov ecx, i

add ecx, 80

invoke GlobalAlloc, 40h, ecx

;invoke HeapAlloc, Hheap, HEAP\_ZERO\_MEMORY, ecx

;invoke VirtualAlloc, 0,ecx,MEM\_RESERVE,PAGE\_READWRITE

mov Hheap, eax

mov ecx, \_start

mov edi, 0

.while(ecx<=\_end)

mov edx, a

mov ebx, Hheap

mov eax, dword ptr [edx+ecx\*4]

mov dword ptr [ebx+edi\*4], eax

inc ecx

inc edi

.endw

mov eax, \_start

mov i, eax

mov swapped, 1

mov eax, \_end

sub eax, \_start

mov indexOfLastUnsortedElement, eax

;sub indexOfLastUnsortedElement, 1

.while(swapped==1)

mov swapped, 0

mov ecx, 0

.while(ecx<indexOfLastUnsortedElement)

mov edi, Hheap

mov eax, dword ptr [edi+ecx\*4]

mov ebx, dword ptr [edi+ecx\*4+4]

cmp eax, ebx

jl iflabel; .if(eax > ebx)

mov esi, dword ptr [edi+ecx\*4+4]

mov dword ptr [edi+ecx\*4], esi

mov dword ptr [edi+ecx\*4+4], eax

mov swapped, 1

iflabel:;.endif

inc ecx

.endw

dec indexOfLastUnsortedElement

.endw

mov eax, Hheap

mov ebx, res

mov dword ptr [ebx], eax

popa

ret

sort endp

2.

* Для успешной компиляции в IDE CLion следует выполнить следующий алгоритм:

|  |  |
| --- | --- |
| 1 | Скомпилировать asm библиотеку |
| 2 | В файл CMakeLists.txt, добавить строку “target\_link\_libraries(untitled7 C://masm32//sampledll.lib)” ! |
| 3 | Добавить Dll библиотеку в папку с нашим исполняемым файлом |

* Для успешной компиляции в IDE Visual Studio следует выполнить следующий алгоритм:

|  |  |
| --- | --- |
| 1 | Скомпилировать asm библиотеку |
| 2 | Скомпилировать программу через командную строчку следующим образом:  cl /FaTest.asm Test.cpp sampledll.lib |
| 3 | Добавить Dll библиотеку в папку с нашим исполняемым файлом |

* + mydll.def

LIBRARY mydll

EXPORTS sort\_Cdecl

EXPORTS sort\_stdcall

EXPORTS sort\_fastcall

EXPORTS sort\_Cdecl\_undefined

EXPORTS sort\_stdcall\_undefined

EXPORTS sort\_CdeclStatic

EXPORTS sort\_stdcall\_static

* + mydll.asm

; ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤

include \masm32\include\masm32rt.inc

; ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤

; -------------------------------------------

; Build this DLL with the provided MAKEIT.BAT

; -------------------------------------------

.data?

hInstance dd ?

.code

; «««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««

LibMain proc instance:DWORD,reason:DWORD,unused:DWORD

.if reason == DLL\_PROCESS\_ATTACH

;mrm hInstance, instance ; copy local to global

mov eax, TRUE ; return TRUE so DLL will start

.elseif reason == DLL\_PROCESS\_DETACH

.elseif reason == DLL\_THREAD\_ATTACH

.elseif reason == DLL\_THREAD\_DETACH

.endif

ret

LibMain endp

sort proc a:dword, \_start:dword, \_end:dword, res:dword

local Hheap:dword, i:dword, indexOfLastUnsortedElement:dword, swapped:dword

pusha

finit

mov ecx, \_end

sub ecx, \_start

mov i, ecx

fild i

mov i, 4

fimul i

fistp i

mov ecx, i

;add ecx, 100

;invoke HeapCreate, 0, ecx, ecx

;mov Hheap, eax

mov ecx, i

add ecx, 80

;invoke GlobalAlloc, 40h, ecx

;invoke HeapAlloc, Hheap, HEAP\_ZERO\_MEMORY, ecx

;invoke VirtualAlloc, 0,ecx,MEM\_RESERVE,PAGE\_READWRITE

mov Hheap, eax

mov ecx, res

mov Hheap, ecx

mov ecx, \_start

mov edi, 0

.while(ecx<=\_end)

mov edx, a

mov ebx, Hheap

mov eax, dword ptr [edx+ecx\*4]

mov dword ptr [ebx+edi\*4], eax

inc ecx

inc edi

.endw

mov eax, \_start

mov i, eax

mov swapped, 1

mov eax, \_end

sub eax, \_start

mov indexOfLastUnsortedElement, eax

;sub indexOfLastUnsortedElement, 1

.while(swapped==1)

mov swapped, 0

mov ecx, 0

.while(ecx<indexOfLastUnsortedElement)

mov edi, Hheap

mov eax, dword ptr [edi+ecx\*4]

mov ebx, dword ptr [edi+ecx\*4+4]

cmp eax, ebx

jl iflabel; .if(eax > ebx)

mov esi, dword ptr [edi+ecx\*4+4]

mov dword ptr [edi+ecx\*4], esi

mov dword ptr [edi+ecx\*4+4], eax

mov swapped, 1

iflabel:;.endif

inc ecx

.endw

dec indexOfLastUnsortedElement

.endw

mov eax, Hheap

mov ebx, res

;mov dword ptr [ebx], eax

popa

ret

sort endp

sort\_Cdecl proc C a:dword, \_start:dword, \_end:dword, res:dword

invoke sort, a, \_start, \_end, res

ret

sort\_Cdecl endp

sort\_stdcall proc stdcall a:dword, \_start:dword, \_end:dword, res:dword

invoke sort, a, \_start, \_end, res

ret

sort\_stdcall endp

sort\_fastcall proc

invoke sort, ecx, edx, dword ptr [esp+8], eax

ret

sort\_fastcall endp

sort\_Cdecl\_undefined proc C

push ebp

mov ebp,esp

push dword ptr ss:[ebp+14h]

push dword ptr ss:[ebp+10h]

push dword ptr ss:[ebp+0Ch]

push dword ptr ss:[ebp+8h]

call sort

leave

ret

sort\_Cdecl\_undefined endp

sort\_stdcall\_undefined proc stdcall

push ebp

mov ebp,esp

push dword ptr ss:[ebp+14h]

push dword ptr ss:[ebp+10h]

push dword ptr ss:[ebp+0Ch]

push dword ptr ss:[ebp+8h]

call sort

leave

ret 10h

sort\_stdcall\_undefined endp

sort\_CdeclStatic proc C a:dword, \_start:dword, \_end:dword, res:dword

invoke sort, a, \_start, \_end, res

ret

sort\_CdeclStatic endp

sort\_stdcall\_static proc stdcall a:dword, \_start:dword, \_end:dword, res:dword

invoke sort, a, \_start, \_end, res

ret

sort\_stdcall\_static endp

end LibMain

3.

#include **<iostream>**#include **<windows.h>**#include **<time.h>***//Статическое подключение dll-библиотеки*#pragma comment(lib, **"C:\\masm32\\mydll.lib"**)  
  
**extern "C" \_\_declspec**(dllimport) **void \_cdecl** sort\_CdeclStatic(**void** \*a, **int** start, **int** b, **void** \*res);  
**extern "C" \_\_declspec**(dllimport) **void \_stdcall** sort\_stdcall\_static(**void** \*a, **int** start, **int** b, **void** \*res);  
  
**void** swap(**int** \*xp, **int** \*yp) {  
 **int** temp = \*xp;  
 \*xp = \*yp;  
 \*yp = temp;  
}  
  
**void** bubbleSort(**int** \*arr, **int** n) {  
 **int** i, j;  
 **for** (i = 0; i<n-1; i++) **for**(j = 0; j<n-i-1; j++) **if** (arr[j] > arr[j + 1]) swap(&(arr[j]), &(arr[j + 1]));  
}  
  
**int** main() {  
 **int** a[] = {1, 2, 3, 4, -1, 5, 4, 3, 2, 1}, \*res; *//Динамическое подключение dll-библиотеки* **typedef int** (**\_cdecl** \*func\_cdecl)(**void** \*a, **int** start, **int** b, **void** \*res);  
 **typedef int** (**\_stdcall** \*func\_stdcall)(**void** \*a, **int** start, **int** b, **void** \*res);  
 **typedef int** (**\_fastcall** \*func\_fastcall)(**void** \*a, **int** start, **int** b, **void** \*res);  
 **typedef int** (**\_cdecl** \*func\_sort\_Cdecl\_undefined)(**void** \*a, **int** start, **int** b, **void** \*res);  
 **typedef int** (**\_stdcall** \*func\_sort\_stdcall\_undefined)(**void** \*a, **int** start, **int** b, **void** \*res);  
  
 **char** dll\_name[] = **"C:\\masm32\\mydll.dll"**; *// Полный путь к библиотеке* HMODULE hModule = LoadLibraryA(dll\_name);  
  
 func\_cdecl f\_Cdecl = (func\_cdecl) GetProcAddress(hModule, **"sort\_Cdecl"**);  
 func\_stdcall f\_stdcall = (func\_stdcall) GetProcAddress(hModule, **"sort\_stdcall"**);  
 func\_fastcall f\_fastcall = (func\_fastcall) GetProcAddress(hModule, **"sort\_fastcall"**);  
 func\_sort\_Cdecl\_undefined f\_Cdeclu = (func\_sort\_Cdecl\_undefined) GetProcAddress(hModule, **"sort\_Cdecl\_undefined"**);  
 func\_sort\_stdcall\_undefined f\_stdcallu = (func\_sort\_stdcall\_undefined) GetProcAddress(hModule,  
 **"sort\_stdcall\_undefined"**);  
  
 res = **static\_cast**<**int** \*>(malloc(4 \* **sizeof**(**int**)));  
  
 sort\_stdcall\_static(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 sort\_CdeclStatic(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 **int** result = f\_Cdecl(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 result = f\_stdcall(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 result = f\_fastcall(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 result = f\_stdcallu(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
 result = f\_Cdeclu(a, 3, 7, res);  
 **for** (**int** i = 0; i <= 4; ++i) printf(**"%i "**, res[i]); puts(**""**);  
  
FreeLibrary(hModule);  
}

4.

#include **<iostream>**

#include **<windef.h>**

#include **<time.h>**

#include **"windows.h"**

**void** swap(**int** \*xp, **int** \*yp)

{

**int** temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

**void** bubbleSort(**int**\* arr, **int** n)

{

**int** i, j;

**for** (i = 0; i < n-1; i++) **for** (j = 0; j < n-i-1; j++) **if** (arr[j] > arr[j+1]) swap(&(arr[j]), &(arr[j+1]));

}

**int** main() {

srand (time(**NULL**));

**int** \*a1, \*a2, \*result, \*result2, N = 5000;

a1 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

a2 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

**for** (**int** i = 0; i < N; ++i) {

**int** randint =rand() % 100 + 1;

a1[i] = randint; a2[i] = randint;

}

**double** t1 = GetTickCount();

bubbleSort(a1,N);

**double** t2 = GetTickCount();

printf (**"%f"**,t2-t1) ;

puts(**""**);

**return** 0;

t1 = GetTickCount();

**\_\_asm\_\_** (**".intel\_syntax noprefix\n"**

**"mov dword ptr ss:[ebp+16],%4\n"**

**"mov dword ptr ss:[ebp+12],%3\n"**

**"mov dword ptr ss:[ebp+20],%5\n"**

**"mov dword ptr ss:[ebp+8],%2\n"**

**"fninit\n"**

**"mov ecx,dword ptr ss:[ebp+16]\n"**

**"sub ecx,dword ptr ss:[ebp+12]\n"**

**"mov dword ptr ss:[ebp-8],ecx\n"**

**"fild dword ptr ss:[ebp-8]\n"**

**"mov dword ptr ss:[ebp-8],4\n"**

**"fimul dword ptr ss:[ebp-8]\n"**

**"fistp dword ptr ss:[ebp-8]\n"**

**"mov ecx,dword ptr ss:[ebp-8]\n"**

**"add ecx,64\n"**

**"push ecx\n"**

**"push ecx\n"**

**"push 0\n"**

**"call %P0\n"**

**"mov dword ptr ss:[ebp-4],eax\n"**

**"mov ecx,dword ptr ss:[ebp-8]\n"**

**"add ecx,50\n"**

**"push ecx\n"**

**"push 8\n"**

**"push dword ptr ss:[ebp-4]\n"**

**"call %P1\n"**

**"mov dword ptr ss:[ebp-4],eax\n"**

**"mov ecx,dword ptr ss:[ebp+12]\n"**

**"mov edi,0\n"**

**"jmp label1\n"**

**"label2:\n"**

**"mov edx,dword ptr ss:[ebp+8]\n"**

**"mov ebx,dword ptr ss:[ebp-4]\n"**

**"mov eax,dword ptr ds:[edx+ecx\*4]\n"**

**"mov dword ptr ds:[ebx+edi\*4],eax\n"**

**"inc ecx\n"**

**"inc edi\n"**

**"label1:\n"**

**"cmp ecx,dword ptr ss:[ebp+16]\n"**

**"jbe label2\n"**

**"mov eax,dword ptr ss:[ebp+12]\n"**

**"mov dword ptr ss:[ebp-8],eax\n"**

**"mov dword ptr ss:[ebp-16],1\n"**

**"mov eax,dword ptr ss:[ebp+16]\n"**

**"sub eax,dword ptr ss:[ebp+12]\n"**

**"mov dword ptr ss:[ebp-12],eax\n"**

**"jmp label3\n"**

**"label3\_2:\n"**

**"mov dword ptr ss:[ebp-16],0\n"**

**"mov ecx,0\n"**

**"jmp label4\n"**

**"label4\_2:\n"**

**"mov edi,dword ptr ss:[ebp-4]\n"**

**"mov eax,dword ptr ds:[edi+ecx\*4]\n"**

**"mov ebx,dword ptr ds:[edi+ecx\*4+4]\n"**

**"cmp eax,ebx\n"**

**"jl label5\n"**

**"mov esi,dword ptr ds:[edi+ecx\*4+4]\n"**

**"mov dword ptr ds:[edi+ecx\*4],esi\n"**

**"mov dword ptr ds:[edi+ecx\*4+4],eax\n"**

**"mov dword ptr ss:[ebp-16],1\n"**

**"label5:\n"**

**"inc ecx\n"**

**"label4:\n"**

**"cmp ecx,dword ptr ss:[ebp-12]\n"**

**"jb label4\_2\n"**

**"dec dword ptr ss:[ebp-12]\n"**

**"label3:\n"**

**"cmp dword ptr ss:[ebp-16],1\n"**

**"je label3\_2\n"**

**"mov eax,dword ptr ss:[ebp-4]\n"**

**"mov ebx,dword ptr ss:[ebp+20]\n"**

**"mov dword ptr ds:[ebx],eax\n"**

**".att\_syntax prefix\n"**

:

:**"i"**(HeapCreate), **"i"**(HeapAlloc), **"r"**(a2), **"r"**(1), **"r"**(N), **"r"**(&result2)

:

);

t2 = GetTickCount();

printf (**"%f\n"**,t2-t1) ;

**return** 0;

}

6 и 7.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | N | DLL (t) | C++ (t) | |  | | --- | |  | |  |  |  |  |  |
| 1 | 5000 | 62 | 281 |  |  |  |  |  |  |
| 2 | 6000 | 47 | 172 |  |  |  |  |  |  |
| 3 | 7000 | 63 | 500 |  |  |  |  |  |  |
| 4 | 8000 | 93 | 485 |  |  |  |  |  |  |
| 5 | 9000 | 109 | 516 |  |  |  |  |  |  |
| 6 | 10000 | 125 | 562 |  |  |  |  |  |  |
| 7 | 11000 | 156 | 563 |  |  |  |  |  |  |
| 8 | 12000 | 203 | 781 |  |  |  |  |  |  |
| 9 | 13000 | 266 | 875 |  |  |  |  |  |  |
| 10 | 14000 | 234 | 860 |  |  |  |  |  |  |

…

srand (time(**NULL**));

**int** \*arr1, \*arr2, \*res2, N = 13000;

arr1 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

arr2 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

res1 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

res2 = **static\_cast**<**int** \*>(malloc(N \* **sizeof**(**int**)));

**for** (**int** i = 0; i < N; ++i) {

**int** randint =rand() % 100 + 1;

arr1[i] = randint; arr2[i] = randint;

}

**double** t1 = GetTickCount();

f\_Cdecl(arr1, 0, N, res);

**double** t2 = GetTickCount();

printf (**"%g\n"**,(t2-t1));

t1 = GetTickCount();

f\_Cdecl(arr2, 0, N, res);

t2 = GetTickCount();

printf (**"%g"**,(t2-t1)) ;

…