**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО**

**ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**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

по теме Способы вызова ассемблерных подпрограмм

в языках высокого уровня

Выполнил: ст. группы ВТ-31  
Новожен Н.В

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.

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.

7 В отчёт включить весь исходный код и графики.

8 Сделать выводы по работе.

**Код lib**

;sum\_int\_2 proc a:DWORD, b:DWORD

;mov EAX, a

;add EAX, b

;ret

;sum\_int\_2 endp

;End LibMain

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include E:\masm32\include\windows.inc

include E:\masm32\include\kernel32.inc

include E:\masm32\include\user32.inc

includelib E:\masm32\lib\user32.lib

includelib E:\masm32\lib\kernel32.lib

include E:\masm32\include\msvcrt.inc

includelib E:\masm32\lib\msvcrt.lib

;.DATA

;a DD 0,2,-1,-8,6,6,6,0,-7,7

;poss DD 10 dup(?)

;negs DD 10 dup(?)

;len DD 10

;poss\_c DD 5

;.CONST

;N dd 16

.CODE

DllMain proc hlnstDLL:DWORD, reason:DWORD, unused:DWORD

mov EAX, 1

ret

DllMain Endp

\_sort proc

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

PUSH EAX

PUSH EBP

XOR EDX,EDX

MOV ESI,[ESP+4\*8]

MOV EBP,[ESP+4\*9]

DEC EBP

FINIT

for1:

INC EDX

MOV ECX,EDX

MOV EDI,EDX

DEC EDI

for2:

CMP ECX,0

JBE j8

FLD DWORD PTR[ESI+ECX\*4];J

FLD DWORD PTR[ESI+EDI\*4];J-1

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JBE j9 ;JBE j9 ;X<=Y

;x[j-1]<=x[j]

;;;;;;SWAP

MOV EAX,[ESI+ECX\*4]

MOV EBX,[ESI+EDI\*4]

MOV [ESI+ECX\*4],EBX

MOV [ESI+EDI\*4],EAX

;;;;;;SWAP

j9:

FFREE ST(0)

DEC ECX

DEC EDI

JMP for2

j8:

CMP EDX,EBP;X<Y

JL for1

POP EBP

POP EAX

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET 8

\_sort endp

\_ssort proc

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

PUSH EAX

PUSH EBP

XOR EDX,EDX

MOV ESI,[ESP+4\*8]

MOV EBP,[ESP+4\*9]

DEC EBP

FINIT

for1:

INC EDX

MOV ECX,EDX

MOV EDI,EDX

DEC EDI

for2:

CMP ECX,0

JBE j8

FLD DWORD PTR[ESI+EDI\*4];J-1

FLD DWORD PTR[ESI+ECX\*4];J

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JBE j9 ;JBE j9 ;X<=Y

;x[j-1]<=x[j]

;;;;;;SWAP

MOV EAX,[ESI+ECX\*4]

MOV EBX,[ESI+EDI\*4]

MOV [ESI+ECX\*4],EBX

MOV [ESI+EDI\*4],EAX

;;;;;;SWAP

j9:

FFREE ST(0)

DEC ECX

DEC EDI

JMP for2

j8:

CMP EDX,EBP;X<Y

JL for1

POP EBP

POP EAX

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET 8

\_ssort endp

;sum\_int\_2c proc stdcall a:DWORD, b:DWORD;

sorts\_arg proc stdcall a1:DWORD, a2:DWORD, a3:DWORD, a4:DWORD, a5:DWORD;

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

MOV ESI,a1;FLOAT POS\_RES

MOV EDI,a2;FLOAT NEG\_RES

;MOV EBX,[ESP+4\*8];INT \* POS\_COUNT

MOV EBX,a4;FLOAT \* A

MOV ECX,a5;INT LENGHT

XOR EDX,EDX

CMP ECX ,0

JLE j1 ;X<=0

FINIT

FLDZ

loop1:

FLD DWORD PTR[EBX+EDX]

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JB j3 ;X<0

;;

PUSH EDI

PUSH ESI

MOV EDI ,EBX

ADD EDI,EDX

MOV EAX ,ESI

MOV ESI,EDI

MOV EDI,EAX

;;

MOVSD [ESI],[EDI];;[EBX+EDX]

POP ESI

POP EDI

ADD ESI,4

JMP j4

j3:

;;

PUSH ESI

MOV ESI ,EBX

ADD ESI,EDX

;;

MOVSD [EDI],[ESI];;[EBX+EDX]

POP ESI

;;ADD EDI,4;;НЕТ ТК EDI CHANGE SELF

j4:

ADD EDX,4

loop loop1

FFREE ST(0)

SUB ESI,a1;FLOAT POS\_RES

SUB EDI,a2;FLOAT NEG\_RES

MOV ECX,4

MOV EAX,ESI

CDQ

DIV ECX

MOV ESI,EAX

MOV EAX,EDI

CDQ

DIV ECX

MOV EDI,EAX

MOV EAX,a3

MOV DWORD PTR [EAX],ESI;INT \* POS\_COUNT

JMP j2

j1:

MOV EDI,0

MOV EAX,a3

MOV DWORD PTR [EAX],EDI;INT \* POS\_COUNT

j2:

MOV EAX,EDI

PUSH ESI

MOV ESI,a1;FLOAT POS\_RES

PUSH ESI

CALL \_sort

PUSH EDI

MOV EDI,a2;FLOAT NEG\_RES

PUSH EDI

CALL \_ssort

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET 20

sorts\_arg endp ;;RET 20

;sum\_int\_2c proc c a:DWORD, b:DWORD; Стиль cdecl

sortc\_arg proc c a1:DWORD, a2:DWORD, a3:DWORD, a4:DWORD, a5:DWORD;

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

MOV ESI,a1;FLOAT POS\_RES

MOV EDI,a2;FLOAT NEG\_RES

;MOV EBX,[ESP+4\*8];INT \* POS\_COUNT

MOV EBX,a4;FLOAT \* A

MOV ECX,a5;INT LENGHT

XOR EDX,EDX

CMP ECX ,0

JLE j1 ;X<=0

FINIT

FLDZ

loop1:

FLD DWORD PTR[EBX+EDX]

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JB j3 ;X<0

;;

PUSH EDI

PUSH ESI

MOV EDI ,EBX

ADD EDI,EDX

MOV EAX ,ESI

MOV ESI,EDI

MOV EDI,EAX

;;

MOVSD [ESI],[EDI];;[EBX+EDX]

POP ESI

POP EDI

ADD ESI,4

JMP j4

j3:

;;

PUSH ESI

MOV ESI ,EBX

ADD ESI,EDX

;;

MOVSD [EDI],[ESI];;[EBX+EDX]

POP ESI

;;ADD EDI,4;;НЕТ ТК EDI CHANGE SELF

j4:

ADD EDX,4

loop loop1

FFREE ST(0)

SUB ESI,a1;FLOAT POS\_RES

SUB EDI,a2;FLOAT NEG\_RES

MOV ECX,4

MOV EAX,ESI

CDQ

DIV ECX

MOV ESI,EAX

MOV EAX,EDI

CDQ

DIV ECX

MOV EDI,EAX

MOV EAX,a3

MOV DWORD PTR [EAX],ESI;INT \* POS\_COUNT

JMP j2

j1:

MOV EDI,0

MOV EAX,a3

MOV DWORD PTR [EAX],EDI;INT \* POS\_COUNT

j2:

MOV EAX,EDI

PUSH ESI

MOV ESI,a1;FLOAT POS\_RES

PUSH ESI

CALL \_sort

PUSH EDI

MOV EDI,a2;FLOAT NEG\_RES

PUSH EDI

CALL \_ssort

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET

sortc\_arg endp ;;RET 20

;sum\_int\_2p proc pascal a:DWORD, b:DWORD; Стиль pascal

sortf proc

; ecx edx

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

push edx

push ecx

MOV ESI,[ESP+4\*6];FLOAT POS\_RES

MOV EDI,[ESP+4\*7];FLOAT NEG\_RES

;MOV EBX,[ESP+4\*8];INT \* POS\_COUNT

MOV EBX,[ESP+4\*9];FLOAT \* A

MOV ECX,[ESP+4\*10];INT LENGHT

XOR EDX,EDX

CMP ECX ,0

JLE j1 ;X<=0

FINIT

FLDZ

loop1:

FLD DWORD PTR[EBX+EDX]

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JB j3 ;X<0

;;

PUSH EDI

PUSH ESI

MOV EDI ,EBX

ADD EDI,EDX

MOV EAX ,ESI

MOV ESI,EDI

MOV EDI,EAX

;;

MOVSD [ESI],[EDI];;[EBX+EDX]

POP ESI

POP EDI

ADD ESI,4

JMP j4

j3:

;;

PUSH ESI

MOV ESI ,EBX

ADD ESI,EDX

;;

MOVSD [EDI],[ESI];;[EBX+EDX]

POP ESI

;;ADD EDI,4;;НЕТ ТК EDI CHANGE SELF

j4:

ADD EDX,4

loop loop1

FFREE ST(0)

SUB ESI,[ESP+4\*6];FLOAT POS\_RES

SUB EDI,[ESP+4\*7];FLOAT NEG\_RES

MOV ECX,4

MOV EAX,ESI

CDQ

DIV ECX

MOV ESI,EAX

MOV EAX,EDI

CDQ

DIV ECX

MOV EDI,EAX

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],ESI;INT \* POS\_COUNT

JMP j2

j1:

MOV EDI,0

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],EDI;INT \* POS\_COUNT

j2:

MOV EAX,EDI

PUSH ESI

MOV ESI,[ESP+4\*6+4];FLOAT POS\_RES

PUSH ESI

CALL \_sort

PUSH EDI

MOV EDI,[ESP+4\*7+4];FLOAT NEG\_RES

PUSH EDI

CALL \_ssort

pop eax

pop edx

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET

sortf endp

;sum\_int\_2c proc c a:DWORD, b:DWORD; Стиль cdecl

sortc proc c

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

MOV ESI,[ESP+4\*6];FLOAT POS\_RES

MOV EDI,[ESP+4\*7];FLOAT NEG\_RES

;MOV EBX,[ESP+4\*8];INT \* POS\_COUNT

MOV EBX,[ESP+4\*9];FLOAT \* A

MOV ECX,[ESP+4\*10];INT LENGHT

XOR EDX,EDX

CMP ECX ,0

JLE j1 ;X<=0

FINIT

FLDZ

loop1:

FLD DWORD PTR[EBX+EDX]

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JB j3 ;X<0

;;

PUSH EDI

PUSH ESI

MOV EDI ,EBX

ADD EDI,EDX

MOV EAX ,ESI

MOV ESI,EDI

MOV EDI,EAX

;;

MOVSD [ESI],[EDI];;[EBX+EDX]

POP ESI

POP EDI

ADD ESI,4

JMP j4

j3:

;;

PUSH ESI

MOV ESI ,EBX

ADD ESI,EDX

;;

MOVSD [EDI],[ESI];;[EBX+EDX]

POP ESI

;;ADD EDI,4;;НЕТ ТК EDI CHANGE SELF

j4:

ADD EDX,4

loop loop1

FFREE ST(0)

SUB ESI,[ESP+4\*6];FLOAT POS\_RES

SUB EDI,[ESP+4\*7];FLOAT NEG\_RES

MOV ECX,4

MOV EAX,ESI

CDQ

DIV ECX

MOV ESI,EAX

MOV EAX,EDI

CDQ

DIV ECX

MOV EDI,EAX

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],ESI;INT \* POS\_COUNT

JMP j2

j1:

MOV EDI,0

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],EDI;INT \* POS\_COUNT

j2:

MOV EAX,EDI

PUSH ESI

MOV ESI,[ESP+4\*6+4];FLOAT POS\_RES

PUSH ESI

CALL \_sort

PUSH EDI

MOV EDI,[ESP+4\*7+4];FLOAT NEG\_RES

PUSH EDI

CALL \_ssort

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET

sortc endp ;;RET 20

;sum\_int\_2s proc stdcall a:DWORD, b:DWORD; Стиль stdcall

sorts proc stdcall

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

MOV ESI,[ESP+4\*6];FLOAT POS\_RES

MOV EDI,[ESP+4\*7];FLOAT NEG\_RES

;MOV EBX,[ESP+4\*8];INT \* POS\_COUNT

MOV EBX,[ESP+4\*9];FLOAT \* A

MOV ECX,[ESP+4\*10];INT LENGHT

XOR EDX,EDX

CMP ECX ,0

JLE j1 ;X<=0

FINIT

FLDZ

loop1:

FLD DWORD PTR[EBX+EDX]

db 0dfh, 0f0h+1;;FCOMIP ST(0),ST(1)

JB j3 ;X<0

;;

PUSH EDI

PUSH ESI

MOV EDI ,EBX

ADD EDI,EDX

MOV EAX ,ESI

MOV ESI,EDI

MOV EDI,EAX

;;

MOVSD [ESI],[EDI];;[EBX+EDX]

POP ESI

POP EDI

ADD ESI,4

JMP j4

j3:

;;

PUSH ESI

MOV ESI ,EBX

ADD ESI,EDX

;;

MOVSD [EDI],[ESI];;[EBX+EDX]

POP ESI

;;ADD EDI,4;;НЕТ ТК EDI CHANGE SELF

j4:

ADD EDX,4

loop loop1

FFREE ST(0)

SUB ESI,[ESP+4\*6];FLOAT POS\_RES

SUB EDI,[ESP+4\*7];FLOAT NEG\_RES

MOV ECX,4

MOV EAX,ESI

CDQ

DIV ECX

MOV ESI,EAX

MOV EAX,EDI

CDQ

DIV ECX

MOV EDI,EAX

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],ESI;INT \* POS\_COUNT

JMP j2

j1:

MOV EDI,0

MOV EAX,[ESP+4\*8]

MOV DWORD PTR [EAX],EDI;INT \* POS\_COUNT

j2:

MOV EAX,EDI

PUSH ESI

MOV ESI,[ESP+4\*6+4];FLOAT POS\_RES

PUSH ESI

CALL \_sort

PUSH EDI

MOV EDI,[ESP+4\*7+4];FLOAT NEG\_RES

PUSH EDI

CALL \_ssort

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

RET 20

sorts endp

;

;START:

; PUSH len

; PUSH offset a

; PUSH offset poss\_c

; PUSH offset negs

; PUSH offset poss

; CALL sorts

; CALL crt\_\_getch

; push NULL

; call ExitProcess

;END START

End DllMain

**КОД lab8.def**

LIBRARY lab8

EXPORTS

sortc

\_sorts@20 =\_sorts@0

@sortf@20 =\_sortf@0

sortc\_arg

sorts\_arg

**КОД lab8.cpp**

// asm8.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <Windows.h>

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

extern "C" \_\_declspec(dllimport) int \_stdcall sorts(float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

extern "C" \_\_declspec(dllimport) int \_cdecl sortc(float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

extern "C" \_\_declspec(dllimport) int \_fastcall sortf(float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

extern "C" \_\_declspec(dllimport) int \_stdcall sorts\_arg(float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

extern "C" \_\_declspec(dllimport) int \_cdecl sortc\_arg(float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

void \_Sort(float\* arr, int n) {

for (int i = 1; i < n; i++) {

for (int j = i; j > 0 && arr[j - 1] > arr[j]; j--) {

float tmp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = tmp;

}

}

}

int Sort(float\* pos, float\* neg, int\* pos\_count, float\* a, int size)

{

\*pos\_count = 0;

int t = 0;

for (size\_t i = 0; i < size; i++)

{

if (a[i] < 0) {

pos[\*pos\_count] = a[i]; \*pos\_count+=1;

}

else {

neg[t] = a[i]; t++;

}

}

\_Sort(pos, \*pos\_count);

\_Sort(neg, t);

return t;

}

int Sort\_asm(float\* pos, float\* neeg, int\* pos\_count, float\* a, int sizze)

{

int t=0;

\_asm{

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

MOV ESI, pos; FLOAT POS\_RES

MOV EDI, neeg; FLOAT NEG\_RES

; MOV EBX, [pos\_count]; INT\* POS\_COUNT

MOV EBX, a; FLOAT\* A

MOV ECX, sizze; INT LENGHT

XOR EDX, EDX

CMP ECX, 0

JLE j1; X <= 0

FINIT

FLDZ

loop1 :

FLD DWORD PTR[EBX + EDX]

FCOMIP ST(0), ST(1); db 0dfh, 0f0h + 1;; FCOMIP ST(0), ST(1)

JB j3; X < 0

;;

PUSH EDI

PUSH ESI

MOV EDI, EBX

ADD EDI, EDX

MOV EAX, ESI

MOV ESI, EDI

MOV EDI, EAX

;;

MOVSD; [ESI] , [EDI];; [EBX + EDX]

POP ESI

POP EDI

ADD ESI, 4

JMP j4

j3 :

;;

PUSH ESI

MOV ESI, EBX

ADD ESI, EDX

;;

MOVSD; [EDI] , [ESI];; [EBX + EDX]

POP ESI

;; ADD EDI, 4;; НЕТ ТК EDI CHANGE SELF

j4 :

ADD EDX, 4

loop loop1

FFREE ST(0)

SUB ESI, pos; FLOAT POS\_RES

SUB EDI, neeg; FLOAT NEG\_RES

MOV ECX, 4

MOV EAX, ESI

CDQ

DIV ECX

MOV ESI, EAX

MOV EAX, EDI

CDQ

DIV ECX

MOV EDI, EAX

MOV EAX, pos\_count

MOV DWORD PTR[EAX], ESI; INT\* POS\_COUNT

JMP j2

j1 :

MOV EDI, 0

MOV EAX, pos\_count

MOV DWORD PTR[EAX], EDI; INT\* POS\_COUNT

j2 :

MOV EAX, EDI

MOV t,EAX

PUSH ESI

MOV ESI, pos; FLOAT POS\_RES

PUSH ESI

////////////////////////////////////////////

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

PUSH EAX

PUSH EBP

XOR EDX, EDX

MOV ESI, [ESP + 4 \* 7]

MOV EBP, [ESP + 4 \* 8]

DEC EBP

FINIT

for1 :

INC EDX

MOV ECX, EDX

MOV EDI, EDX

DEC EDI

for2 :

CMP ECX, 0

JBE j8

FLD DWORD PTR[ESI + ECX \* 4]; J

FLD DWORD PTR[ESI + EDI \* 4]; J - 1

FCOMIP ST(0), ST(1); db 0dfh, 0f0h + 1;; FCOMIP ST(0), ST(1)

JBE j9; JBE j9; X <= Y

; x[j - 1] <= x[j]

;;;;;; SWAP

MOV EAX, [ESI + ECX \* 4]

MOV EBX, [ESI + EDI \* 4]

MOV[ESI + ECX \* 4], EBX

MOV[ESI + EDI \* 4], EAX

;;;;;; SWAP

j9 :

FFREE ST(0)

DEC ECX

DEC EDI

JMP for2

j8 :

CMP EDX, EBP; X < Y

JL for1

POP EBP

POP EAX

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

POP ESI

POP ESI

/////////////////////////////////////////////

PUSH EDI

MOV EDI, neeg; FLOAT NEG\_RES

PUSH EDI

/////////////////////////////////////////////

PUSH ECX

PUSH EBX

PUSH ESI

PUSH EDI

PUSH EDX

PUSH EAX

PUSH EBP

XOR EDX, EDX

MOV ESI, [ESP + 4 \* 7]

MOV EBP, [ESP + 4 \* 8]

DEC EBP

FINIT

for11 :

INC EDX

MOV ECX, EDX

MOV EDI, EDX

DEC EDI

for21 :

CMP ECX, 0

JBE j81

FLD DWORD PTR[ESI + EDI \* 4]; J - 1

FLD DWORD PTR[ESI + ECX \* 4]; J

FCOMIP ST(0), ST(1); db 0dfh, 0f0h + 1;; FCOMIP ST(0), ST(1)

JBE j91; JBE j9; X <= Y

; x[j - 1] <= x[j]

;;;;;; SWAP

MOV EAX, [ESI + ECX \* 4]

MOV EBX, [ESI + EDI \* 4]

MOV[ESI + ECX \* 4], EBX

MOV[ESI + EDI \* 4], EAX

;;;;;; SWAP

j91 :

FFREE ST(0)

DEC ECX

DEC EDI

JMP for21

j81 :

CMP EDX, EBP; X < Y

JL for11

POP EBP

POP EAX

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

POP EDI

POP EDI

/////////////////////////////////////////////

POP EDX

POP EDI

POP ESI

POP EBX

POP ECX

}

return t;

}

int N = 1024;

void show()

{

int\* coun = new int(0);

float\* arr = new float[10];

float\* pos = new float[10];

float\* neg = new float[10];

float arrs[10] = { 0.0,2.0,-1.0,-8.0,6.0,6.0,6.0,0.0,-7.0,7.0 };

for (size\_t i = 0; i < 10; i++)

{

arr[i]=arrs[i] ;

}

ULONGLONG t1, t2;

t1 = GetTickCount64();

int ng = Sort\_asm(pos, neg, coun, arr, 10);//sorts

t2 = GetTickCount64();

t2 = t2 - t1;

std::cout << "test" << "\n";

std::cout << "a=";

for (size\_t i = 0; i < 10; i++)

{

std::cout << arr[i] << " ";

}

std::cout << "\n";

std::cout << "neg=";

for (size\_t i = 0; i < \*coun; i++)

{

std::cout << pos[i] << " ";

}

std::cout << "\n";

std::cout << "pos=";

for (size\_t i = 0; i < ng; i++)

{

std::cout << neg[i] << " ";

}

std::cout << "\n";

std::cout << "time=" << t2 << "\n\n";

}

void fun(std::string s,float\* pos, float\* neg, int\* pos\_count, float\* a, int size)

{

ULONGLONG t1, t2;

t1 = GetTickCount64();

int ng = sorts(pos, neg, pos\_count, a, size);

t2 = GetTickCount64();

t2 = t2 - t1;

std::cout << s << "\n";

/\*std::cout << "a=";

for (size\_t i = 0; i < size; i++)

{

std::cout << a[i] << " ";

}

std::cout << "\n";

std::cout << "neg=";

for (size\_t i = 0; i < \*pos\_count; i++)

{

std::cout << pos[i] << " ";

}

std::cout << "\n";

std::cout << "pos=";

for (size\_t i = 0; i < ng; i++)

{

std::cout << neg[i] << " ";

}\*/

std::cout << "\n";

std::cout << "time=" << t2 << "\n\n";

}

void fun1(std::string s, float\* pos, float\* neg, int\* pos\_count, float\* a, int size)

{

ULONGLONG t1, t2;

t1 = GetTickCount64();

int ng = Sort(pos, neg, pos\_count, a, size);

t2 = GetTickCount64();

t2 = t2 - t1;

std::cout << s << "\n";

std::cout << "\n";

std::cout << "time=" << t2 << "\n\n";

}

void fun2(std::string s, float\* pos, float\* neg, int\* pos\_count, float\* a, int size)

{

ULONGLONG t1, t2;

t1 = GetTickCount64();

int ng = Sort\_asm(pos, neg, pos\_count, a, size);

t2 = GetTickCount64();

t2 = t2 - t1;

std::cout << s << "\n";

std::cout << "\n";

std::cout << "time=" << t2 << "\n\n";

}

void get\_arr(float\* f, int n) {

for (size\_t i = 0; i < n; i++)

{

if(rand()%2==1)f[i] = float(rand());

else f[i] = -float(rand());

}

}

char dll\_name[] = "lab8.dll"; // Полный путь к библиотеке

HMODULE hModule = LoadLibraryA(dll\_name);

typedef int(\_stdcall\* func\_sorts) (float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

typedef int(\_cdecl\* func\_sortc) (float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

typedef int(\_fastcall\* func\_sortf) (float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

typedef int(\_stdcall\* func\_sorts\_arg) (float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

typedef int(\_cdecl\* func\_sortc\_arg) (float\* pos, float\* neg, int\* pos\_count, float\* a, int size);

void SSotr\_dinamik()

{

// Если dll-файл найден

if (hModule != NULL)

{

char func\_name[] = "\_sorts@20";

char func\_name1[] = "sortc";

char func\_name2[] = "@sortf@20";

char func\_name3[] = "sortc\_arg";

char func\_name4[] = "sorts\_arg";

// Получить указатель на подпрограмму

func\_sorts f1 = (func\_sorts)GetProcAddress(hModule, func\_name);

func\_sortc f2 = (func\_sortc)GetProcAddress(hModule, func\_name1);

func\_sortf f3 = (func\_sortf)GetProcAddress(hModule, func\_name2);

func\_sorts\_arg f4 = (func\_sorts\_arg)GetProcAddress(hModule, func\_name3);

func\_sortc\_arg f5 = (func\_sortc\_arg)GetProcAddress(hModule, func\_name4);

if (f1 == NULL)

printf("Подпрограмма %s не найдена в библиотеке", func\_name);

if (f2 == NULL)

printf("Подпрограмма %s не найдена в библиотеке", func\_name1);

if (f3 == NULL)

printf("Подпрограмма %s не найдена в библиотеке", func\_name2);

if (f4 == NULL)

printf("Подпрограмма %s не найдена в библиотеке", func\_name3);

if (f5 == NULL)

printf("Подпрограмма %s не найдена в библиотеке", func\_name4);

if(f1 != NULL&& f2 != NULL&& f3 != NULL&& f4 != NULL&& f5 != NULL)

{

printf("Подпрограмма %s %s %s %s %s найдены в библиотеке и одна из них use", func\_name, func\_name1,func\_name2, func\_name3, func\_name);

int\* coun = new int(0);

float\* arr = new float[10];

float\* pos = new float[10];

float\* neg = new float[10];

float arrs[10] = { 0.0,2.0,-1.0,-8.0,6.0,6.0,6.0,0.0,-7.0,7.0 };

for (size\_t i = 0; i < 10; i++)

{

arr[i] = arrs[i];

}

int ng = f1(pos, neg, coun, arr, 10);//sorts

std::cout << "test" << "\n";

std::cout << "a=";

for (size\_t i = 0; i < 10; i++)

{

std::cout << arr[i] << " ";

}

std::cout << "\n";

std::cout << "neg=";

for (size\_t i = 0; i < \*coun; i++)

{

std::cout << pos[i] << " ";

}

std::cout << "\n";

std::cout << "pos=";

for (size\_t i = 0; i < ng; i++)

{

std::cout << neg[i] << " ";

}

std::cout << "\n";

delete[] arr;

delete[] pos;

delete[] neg;

delete coun;

}

// Освобождение библиотеки

FreeLibrary(hModule);

}

else

printf("dll-файл %s не найден", dll\_name);

}

int main()

{

setlocale(LC\_ALL, "Russian");

int ng = 0;

N = N \* 10;

int coun;

float \*arr = new float[N];

float \*pos = new float[N];

float \*neg = new float[N];

show();

get\_arr(arr, N);

fun("sort stdcall ", pos, neg, &coun, arr, N);

fun1("sort c++ ", pos, neg, &coun, arr, N);

fun2("sort c++ ", pos, neg, &coun, arr, N);

SSotr\_dinamik();

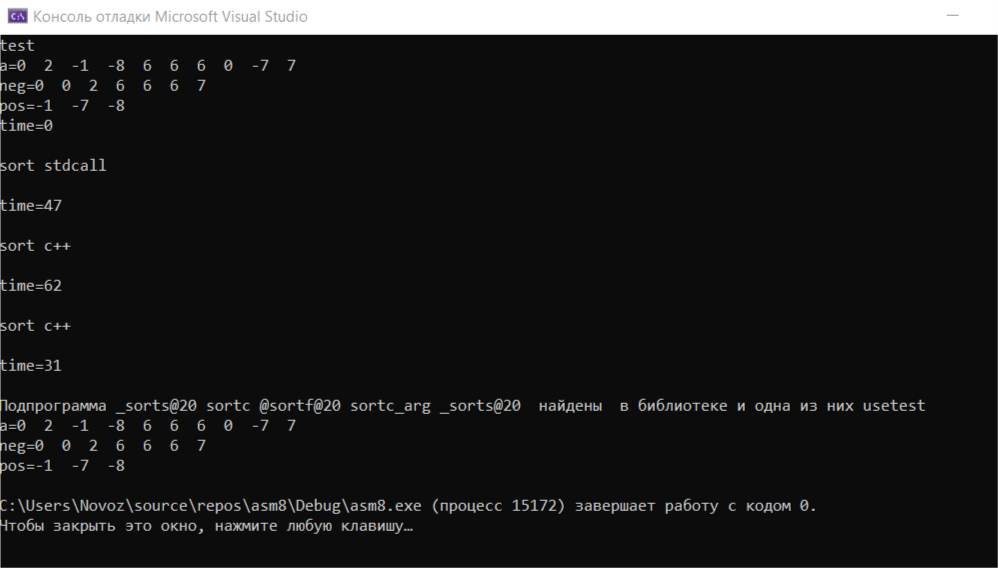
delete[] arr;

delete[] pos;

delete[] neg;

//fun2("asm c++ ", pos, neg, coun, arr, N);

}



Вывод: мы изучили способы вызова подпрограмм(в стилях stdcall, cdecl, fastcall, без явного перечисления аргументов в заголовке и в стилях stdcall, cdecl, написанные с перечислением аргументов в заголовке подпрограммы), написанных на разных языках программирования посредством dll-библиотек. Подключать все подпрограммы из dll-библиотеки к проектам на C# и С++ статическим и динамическим способом. Использовали ассемблерую вставку языка C++.Использовал API-функции GetTickCount().