**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-32  
Воскобойников И. С.

Проверил: Осипов.О.В

**Белгород 2020**

**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.

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.

.386 ; Тип процессора

.model flat, stdcall ; Модель памяти и стиль вызова подпрограмм

option casemap: none ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

; --- Подключаемые библиотеки ---

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.CODE

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

MOV EAX, 1

RET

LibMain Endp

; int search\_min\_with\_i (int\* a, int n, int i) - возвращает номер минимального элемента в массиве a размера n, найденного, начиная с позиции i

search\_min\_with\_i proc stdcall ; Префикс меток - smwi

;20

PUSHF

PUSHAD

MOV EDI, [ESP + 38]

MOV ECX, [ESP + 42]

MOV EAX, [ESP + 46]

MOV EBP, EAX

MOV ESI, EAX

SUB ECX, EAX

MOV EBX, 4h

IMUL EBX

ADD EDI, EAX

MOV EAX, 7FFFFFFFh

smwi\_again:

CMP DWORD PTR [EDI], EAX

JL smwi\_again\_new\_min

JMP smwi\_again\_next

smwi\_again\_new\_min:

MOV EAX, DWORD PTR [EDI]

MOV ESI, EBP

JMP smwi\_again\_next

smwi\_again\_next:

ADD EDI, 4h

INC EBP

LOOP smwi\_again

MOV DWORD PTR [ESP + 28], ESI

POPAD

POPF

RET 12

search\_min\_with\_i endp

; void sort\_change (int\* a, int n) - Сортирует выбором массив a размера n по неубыванию

sort\_change proc stdcall

PUSHF

PUSHAD

MOV EDI, [ESP + 38]

MOV ECX, [ESP + 42]

MOV EDX, ECX

MOV ESI, 0h

sch\_again:

MOV EBX, DWORD PTR [EDI + 4\*ESI]

PUSH ESI

PUSH EDX

PUSH EDI

CALL search\_min\_with\_i

MOV EBP, DWORD PTR [EDI + 4\*EAX]

MOV DWORD PTR [EDI + 4\*ESI], EBP

MOV DWORD PTR [EDI + 4\*EAX], EBX

INC ESI

LOOP sch\_again

POPAD

POPF

RET 8

sort\_change endp

; void output\_array (int\* a, int n)

output\_array proc stdcall

PUSHF

PUSHAD

MOV EBP, ESP

SUB ESP, 4h

MOV BYTE PTR [ESP], '%'

MOV BYTE PTR [ESP + 1], 'i'

MOV BYTE PTR [ESP + 2], ' '

MOV BYTE PTR [ESP + 3], 0h

MOV EDI, [EBP + 38]

MOV ECX, [EBP + 42]

outarr\_again:

PUSH ECX

PUSH DWORD PTR [EDI]

SUB EBP, 4h

PUSH EBP

ADD EBP, 4h

CALL crt\_printf

ADD ESP, 8h

POP ECX

ADD EDI, 4h

LOOP outarr\_again

ADD ESP, 4h

POPAD

POPF

RET 8

output\_array endp

;126

; int to\_pos\_neg\_array (int\* neg\_res, int\* pos\_res, int\* neg\_count, int\* a, int length)

to\_pos\_neg\_array\_1s proc stdcall

PUSHF

PUSHAD

MOV EDI, [ESP + 38]

MOV ESI, [ESP + 42]

MOV EBP, [ESP + 50]

MOV ECX, [ESP + 54]

MOV EAX, 0

pos\_count

MOV EBX, 0

neg\_count

topna1s\_again:

MOV EDX, DWORD PTR [EBP]

CMP EDX, 0

JL topna1s\_again\_neg

MOV DWORD PTR [ESI], EDX

ADD ESI, 4h

INC EAX

JMP topna1s\_again\_next

topna1s\_again\_neg:

MOV DWORD PTR [EDI], EDX

ADD EDI, 4h

INC EBX

JMP topna1s\_again\_next

topna1s\_again\_next:

ADD EBP, 4h

LOOP topna1s\_again

PUSH DWORD PTR [ESP + 42]

CALL sort\_change

PUSH EAX

PUSH DWORD PTR [ESP + 46]

CALL sort\_change

;170

MOV EBP, [ESP + 46]

neg\_count

MOV DWORD PTR [EBP], EBX

neg\_count

MOV DWORD PTR [ESP + 28], EAX

POPAD

POPF

RET 20

to\_pos\_neg\_array\_1s endp

to\_pos\_neg\_array\_1c proc c

topna1c

PUSHF

PUSHAD

MOV EDI, [ESP + 38]

MOV ESI, [ESP + 42]

MOV EBP, [ESP + 50]

MOV ECX, [ESP + 54]

MOV EAX, 0

pos\_count

MOV EBX, 0

neg\_count

topna1c\_again:

MOV EDX, DWORD PTR [EBP]

CMP EDX, 0 ; Если число отрицательное, то сохраняем его в neg\_res и увеличиваем neg\_count

JL topna1c\_again\_neg

MOV DWORD PTR [ESI], EDX

ADD ESI, 4h

INC EAX

JMP topna1c\_again\_next

topna1c\_again\_neg:

MOV DWORD PTR [EDI], EDX

ADD EDI, 4h

INC EBX

JMP topna1c\_again\_next

topna1c\_again\_next:

ADD EBP, 4h

LOOP topna1c\_again

PUSH EBX

PUSH DWORD PTR [ESP + 42]

CALL sort\_change

;217

PUSH EAX

PUSH DWORD PTR [ESP + 46]

CALL sort\_change

MOV EBP, [ESP + 46]

neg\_count

MOV DWORD PTR [EBP], EBX

neg\_count

MOV DWORD PTR [ESP + 28], EAX

POPAD

POPF

RET

to\_pos\_neg\_array\_1c endp

to\_pos\_neg\_array\_1f proc fastcall

topna1f

PUSHF

PUSHAD

MOV EDI, ECX

MOV ESI, EDX

MOV EDX, [EBP + 12]

MOV ECX, [EBP + 16]

MOV EAX, 0

pos\_count

MOV EBX, 0

neg\_count

topna1f\_again:

PUSH EAX

MOV EAX, DWORD PTR [EDX]

CMP EAX, 0

JL topna1f\_again\_neg

MOV DWORD PTR [ESI], EAX

ADD ESI, 4h

POP EAX

INC EAX

JMP topna1f\_again\_next

topna1f\_again\_neg:

MOV DWORD PTR [EDI], EAX

ADD EDI, 4h

POP EAX

INC EBX

JMP topna1f\_again\_next

topna1f\_again\_next:

ADD EDX, 4h

LOOP topna1f\_again

PUSH EBX

PUSH DWORD PTR [ESP + 28]

CALL sort\_change

PUSH EAX

PUSH DWORD PTR [ESP + 24]

CALL sort\_change

MOV EDX, [EBP + 8]

MOV DWORD PTR [EDX], EBX

MOV DWORD PTR [ESP + 28], EAX POPAD

POPF

RET 20

to\_pos\_neg\_array\_1f endp

;284

to\_pos\_neg\_array\_2s proc stdcall neg\_res:DWORD, pos\_res:DWORD, neg\_count:DWORD, a:DWORD, l:DWORD

PUSHF

PUSHAD

MOV EDI, neg\_res

MOV ESI, pos\_res

MOV EDX, a

MOV ECX, l

MOV EAX, 0

MOV EBX, 0

topna2s\_again:

PUSH EAX

MOV EAX, DWORD PTR [EDX]

CMP EAX, 0

JL topna2s\_again\_neg

MOV DWORD PTR [ESI], EAX

ADD ESI, 4h

POP EAX

INC EAX

JMP topna2s\_again\_next

topna2s\_again\_neg:

MOV DWORD PTR [EDI], EAX

ADD EDI, 4h

POP EAX

INC EBX

JMP topna2s\_again\_next

topna2s\_again\_next:

ADD EDX, 4h

LOOP topna2s\_again

PUSH EBX

PUSH neg\_res

CALL sort\_change

PUSH EAX

PUSH pos\_res

CALL sort\_change

MOV EDX, neg\_count

MOV DWORD PTR [EDX], EBX

MOV DWORD PTR [ESP + 28], EAX

POPAD

POPF

RET 20

to\_pos\_neg\_array\_2s endp

to\_pos\_neg\_array\_2c proc c neg\_res:DWORD, pos\_res:DWORD, neg\_count:DWORD, a:DWORD, l:DWORD

PUSHF

PUSHAD

MOV EDI, neg\_res

MOV ESI, pos\_res

MOV EDX, a

MOV ECX, l

MOV EAX, 0

MOV EBX, 0

topna2c\_again:

PUSH EAX

MOV EAX, DWORD PTR [EDX]

CMP EAX, 0

JL topna2c\_again\_neg

MOV DWORD PTR [ESI], EAX

ADD ESI, 4h

POP EAX

INC EAX

JMP topna2c\_again\_next

topna2c\_again\_neg:

MOV DWORD PTR [EDI], EAX

ADD EDI, 4h

POP EAX

INC EBX

JMP topna2c\_again\_next

topna2c\_again\_next:

ADD EDX, 4h

LOOP topna2c\_again

PUSH EBX

PUSH neg\_res

CALL sort\_change

PUSH EAX

PUSH pos\_res

CALL sort\_change

MOV EDX, neg\_count

MOV DWORD PTR [EDX], EBX

MOV DWORD PTR [ESP + 28], EAX

POPAD

POPF

RET

to\_pos\_neg\_array\_2c endp

END LibMain

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.

Статический способ

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

#include <iostream>

extern "C" \_\_declspec(dllimport) void \_stdcall output\_array(int\*, int);

extern "C" \_\_declspec(dllimport) int \_stdcall to\_pos\_neg\_array\_1s(int\*, int\*, int\*, int\*, int);

extern "C" \_\_declspec(dllimport) int \_stdcall to\_pos\_neg\_array\_2s(int\*, int\*, int\*, int\*, int);

extern "C" \_\_declspec(dllimport) int \_cdecl to\_pos\_neg\_array\_1c(int\*, int\*, int\*, int\*, int);

extern "C" \_\_declspec(dllimport) int \_cdecl to\_pos\_neg\_array\_2c(int\*, int\*, int\*, int\*, int);

extern "C" \_\_declspec(dllimport) int \_fastcall to\_pos\_neg\_array\_1f(int\*, int\*, int\*, int\*, int);

int main()

{

int length = 8, neg\_count, pos\_count;

int a[8] = {5, 14, -3, -2, 1, 2, -77, 16}, neg\_res[8], pos\_res[8];

pos\_count = to\_pos\_neg\_array\_1c(neg\_res, pos\_res, &neg\_count, a, length);

output\_array(neg\_res, neg\_count);

std::cout << std::endl;

output\_array(pos\_res, pos\_count);

return 0;

}

Динамический способ

#include <iostream>

#include <windows.h>

typedef int(\_stdcall\* func1) (int\*, int\*, int\*, int\*, int);

typedef void(\_stdcall\* func2) (int\*, int);

int main()

{

int length = 8, neg\_count, pos\_count;

int a[8] = { -1, 4, -17, 224, 2, 1, -4, -88 }, neg\_res[8], pos\_res[8];

HMODULE hModule = LoadLibraryA("E:\\C\_Project\\asm\_lab8\_dynamic\\lab8lib.dll");

if (hModule != NULL)

{

func1 f1 = (func1) GetProcAddress(hModule, "\_to\_pos\_neg\_array\_1s@20");

func2 f2 = (func2) GetProcAddress(hModule, "\_output\_array@8");

if (f1 == NULL)

printf("No function 1\n");

if (f2 == NULL)

printf("No function 2\n");

else

{

pos\_count = f1(neg\_res, pos\_res, &neg\_count, a, length);

f2(neg\_res, neg\_count);

std::cout << std::endl;

f2(pos\_res, pos\_count);

}

FreeLibrary(hModule);

}

else

printf("No dll");

return 0;

}

4. Написать подпрограмму для решения задачи варианта с использованием ассемблерной вставки на языке C++.

#include <iostream>

void output\_array(int\* a, int n)

{

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

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

}

void swap(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

int search\_min\_with\_i(int\* a, int n, int i)

{

int min = a[i];

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

{

if (a[j] < min)

{

min = a[j];

i = j;

}

}

return i;

}

void (\_stdcall sort\_change) (int\* a, int n)

{

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

{

int j = search\_min\_with\_i(a, n, i);

swap(&a[i], &a[j]);

}

}

int (\_stdcall to\_pos\_neg\_array\_asm) (int\* neg\_res, int\* pos\_res, int\* neg\_count, int\* a, int len)

{

int buf;

\_asm

{

MOV EDI, neg\_res; Достали neg\_res

MOV ESI, pos\_res; Достали pos\_res

MOV EDX, a; Достали a

MOV ECX, len; Достали length

MOV EAX, 0; Здесь будем считать pos\_count

MOV EBX, 0; Здесь будем считать neg\_count

topna1s\_again :

PUSH EAX

MOV EAX, DWORD PTR[EDX]

CMP EAX, 0; Если число отрицательное, то сохраняем его в neg\_res и увеличиваем neg\_count

JL topna1s\_again\_neg

MOV DWORD PTR[ESI], EAX; Иначе сохраняем его в pos\_res и увеличиваем pos\_count

ADD ESI, 4h

POP EAX

INC EAX

JMP topna1s\_again\_next

topna1s\_again\_neg:

MOV DWORD PTR[EDI], EAX

ADD EDI, 4h

POP EAX

INC EBX

JMP topna1s\_again\_next

topna1s\_again\_next :

ADD EDX, 4h

LOOP topna1s\_again

PUSH EBX; Сортируем массив отрицательных чисел

PUSH neg\_res

CALL sort\_change

PUSH EAX; Сортируем массив положительных чисел

PUSH pos\_res

CALL sort\_change

MOV EDX, neg\_count; Достали адрес neg\_count

MOV DWORD PTR[EDX], EBX; Записали neg\_count

MOV buf, EAX; Сохранили pos\_count в буфер

}

return buf;

}

int main()

{

int length = 8, neg\_count, pos\_count;

int a[8] = { 12, 52, -4, -51, 13, -1, -2, 3333 }, neg\_res[8], pos\_res[8];

pos\_count = to\_pos\_neg\_array\_asm(neg\_res, pos\_res, &neg\_count, a, length);

output\_array(neg\_res, neg\_count);

std::cout << std::endl;

output\_array(pos\_res, pos\_count);

return 0;

}

5. Написать подпрограммы для решения задачи варианта с использованием обычного высокоуровнего языка C# и C++ (или любого другого).

int to\_pos\_neg\_array\_cpp (int\* neg\_res, int\* pos\_res, int\* neg\_count, int\* a, int len)

{

\*neg\_count = 0;

int pos\_count = 0;

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

{

if (a[i] < 0)

neg\_res[\*neg\_count] = a[i], (\*neg\_count)++;

else

pos\_res[pos\_count] = a[i], pos\_count++;

}

sort\_change(neg\_res, \*neg\_count);

sort\_change(pos\_res, pos\_count);

return pos\_count;

}

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 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().

**Примечание**: asm – алгоритм на ассемблере, asm\_cpp – на ассемблерной вставке, cpp – на С++

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.

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

#include <iostream>

#include <random>

#include <ctime>

#include <Windows.h>

extern "C" \_\_declspec(dllimport) void \_stdcall output\_array(int\*, int);

extern "C" \_\_declspec(dllimport) int \_stdcall to\_pos\_neg\_array\_1s(int\*, int\*, int\*, int\*, int);

void swapb(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

int search\_min\_with\_ib(int\* a, int n, int i)

{

int min = a[i];

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

{

if (a[j] < min)

{

min = a[j];

i = j;

}

}

return i;

}

void(\_stdcall sort\_changes) (int\* a, int n)

{

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

{

int j = search\_min\_with\_ib(a, n, i);

swapb(&a[i], &a[j]);

}

}

int (\_stdcall to\_pos\_neg\_array\_asm) (int\* neg\_res, int\* pos\_res, int\* neg\_count, int\* a, int len)

{

int buf;

\_asm

{

MOV EDI, neg\_res; Достали neg\_res

MOV ESI, pos\_res; Достали pos\_res

MOV EDX, a; Достали a

MOV ECX, len; Достали length

MOV EAX, 0; Здесь будем считать pos\_count

MOV EBX, 0; Здесь будем считать neg\_count

topna1s\_again :

PUSH EAX

MOV EAX, DWORD PTR[EDX]

CMP EAX, 0; Если число отрицательное, то сохраняем его в neg\_res и увеличиваем neg\_count

JL topna1s\_again\_neg

MOV DWORD PTR[ESI], EAX; Иначе сохраняем его в pos\_res и увеличиваем pos\_count

ADD ESI, 4h

POP EAX

INC EAX

JMP topna1s\_again\_next

topna1s\_again\_neg :

MOV DWORD PTR[EDI], EAX

ADD EDI, 4h

POP EAX

INC EBX

JMP topna1s\_again\_next

topna1s\_again\_next :

ADD EDX, 4h

LOOP topna1s\_again

PUSH EBX; Сортируем массив отрицательных чисел

PUSH neg\_res

CALL sort\_changes

PUSH EAX; Сортируем массив положительных чисел

PUSH pos\_res

CALL sort\_changes

MOV EDX, neg\_count; Достали адрес neg\_count

MOV DWORD PTR[EDX], EBX; Записали neg\_count

MOV buf, EAX; Сохранили pos\_count в буфер

}

return buf;

}

int to\_pos\_neg\_array\_cpp(int\* neg\_res, int\* pos\_res, int\* neg\_count, int\* a, int len)

{

\*neg\_count = 0;

int pos\_count = 0;

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

{

if (a[i] < 0)

neg\_res[\*neg\_count] = a[i], (\*neg\_count)++;

else

pos\_res[pos\_count] = a[i], pos\_count++;

}

sort\_changes(neg\_res, \*neg\_count);

sort\_changes(pos\_res, pos\_count);

return pos\_count;

}

void rand\_array(int\* a, int n)

{

srand(static\_cast<unsigned int>(time(0)));

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

a[i] = rand() \* (int) pow(-1, rand() % 2);

}

const int length = 100000;

int main()

{

int neg\_count, pos\_count;

int\* a = new int[length], \* neg\_res = new int[length], \* pos\_res = new int[length];

rand\_array(a, length);

DWORD start = GetTickCount();

pos\_count = to\_pos\_neg\_array\_1s(neg\_res, pos\_res, &neg\_count, a, length);

std::cout << "Asm: " << GetTickCount() - start << std::endl;

start = GetTickCount();

pos\_count = to\_pos\_neg\_array\_asm(neg\_res, pos\_res, &neg\_count, a, length);

std::cout << "Asm\_cpp: " << GetTickCount() - start << std::endl;

start = GetTickCount();

pos\_count = to\_pos\_neg\_array\_cpp(neg\_res, pos\_res, &neg\_count, a, length);

std::cout << "Cpp: " << GetTickCount() - start << std::endl;

delete[] a;

delete[] neg\_res;

delete[] pos\_res;

return 0;

}

8. Сделать выводы по работе

В ходе выполнения работы я научился создавать и использовать dll-библиотеки, а также сравнил скорость работы подпрограмм, написанных на ассемблере, С++ и ассемблерных вставках в С++. При небольших тестовых данных лучше работают алгоритмы, написанные на чистом ассемблере, а при больших – на языке С++.