федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»

Направление подготовки 09.03.04 «Программная инженерия»

Дисциплина «Электротехника»

**Лабораторная РАБОТА №6**

**Ассемблерные вставки в программах на C++. Целочисленная арифметика и арифметика с плавающей запятой**

Работу выполнил студенты группы ПИН-24 Баранов Д.А. и Демочкина А.В.

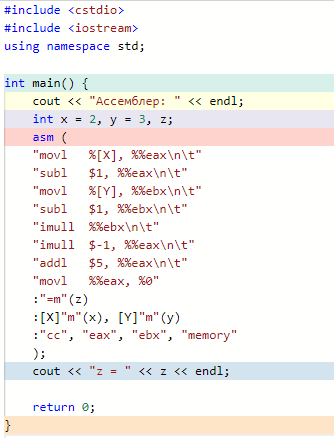
Работу проверил ассистент Института СПИНТех Фомин Р.А.

**Цель работы:** научиться использовать базовые команды x86 и команды расширений AVX/SSE.

**Задание 1.**

Вычислите для заданных целых 𝑥 и 𝑦:





#include <cstdio>

#include <iostream>

using namespace std;

int main() {

cout << "Ассемблер: " << endl;

int x = 2, y = 3, z;

asm (

"movl %[X], %%eax\n\t"

"subl $1, %%eax\n\t"

"movl %[Y], %%ebx\n\t"

"subl $1, %%ebx\n\t"

"imull %%ebx\n\t"

"imull $-1, %%eax\n\t"

"addl $5, %%eax\n\t"

"movl %%eax, %0"

:"=m"(z)

:[X]"m"(x), [Y]"m"(y)

:"cc", "eax", "ebx", "memory"

);

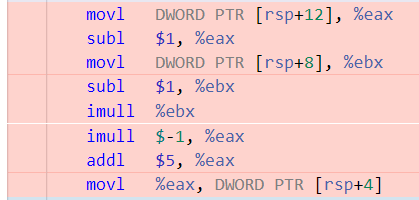
cout << "z = " << z << endl;

return 0;

}

Ассемблер:

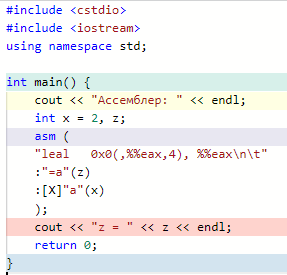
z = 3



**Задание 2.**

Вычислите для заданного целого 𝑥:





#include <cstdio>

#include <iostream>

using namespace std;

int main() {

cout << "Ассемблер: " << endl;

int x = -5, z;

asm (

"leal 0x0(,%%eax,4), %%eax\n\t"

:"=a"(z)

:[X]"a"(x)

);

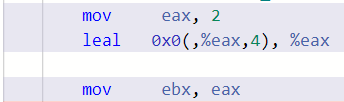
cout << "z = " << z << endl;

return 0;

}

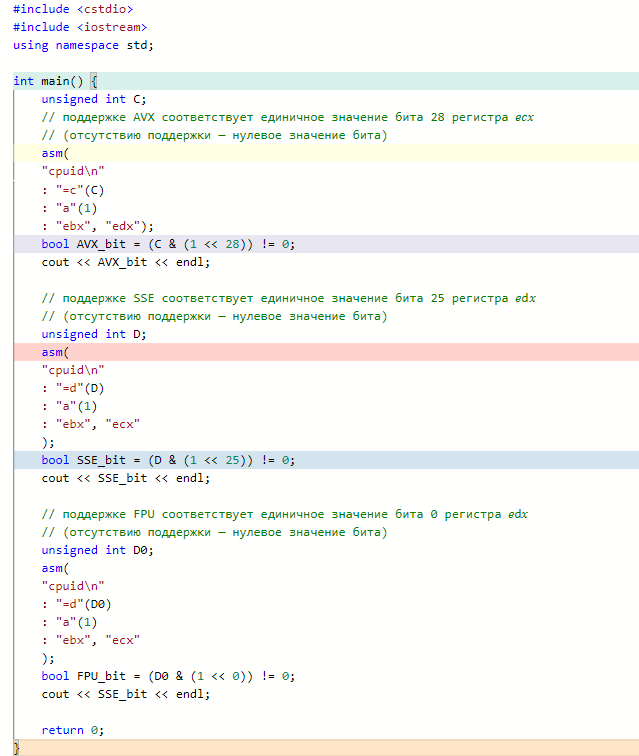
Ассемблер:

z = 8



**Задание 3.**

Проверьте, доступны ли на используемой платформе команды AVX, SSE, FPU.

****

**#include <cstdio>**

**#include <iostream>**

**using namespace std;**

**int main() {**

**unsigned int C;**

**// поддержке AVX соответствует единичное значение бита 28 регистра 𝑒𝑐𝑥**

**// (отсутствию поддержки — нулевое значение бита)**

**asm(**

**"cpuid\n"**

**: "=c"(C)**

**: "a"(1)**

**: "ebx", "edx");**

**bool AVX\_bit = (C & (1 << 28)) != 0;**

**cout << AVX\_bit << endl;**

**// поддержке SSE соответствует единичное значение бита 25 регистра 𝑒d𝑥**

**// (отсутствию поддержки — нулевое значение бита)**

**unsigned int D;**

**asm(**

**"cpuid\n"**

**: "=d"(D)**

**: "a"(1)**

**: "ebx", "ecx"**

**);**

**bool SSE\_bit = (D & (1 << 25)) != 0;**

**cout << SSE\_bit << endl;**

**// поддержке FPU соответствует единичное значение бита 0 регистра 𝑒d𝑥**

**// (отсутствию поддержки — нулевое значение бита)**

**unsigned int D0;**

**asm(**

**"cpuid\n"**

**: "=d"(D0)**

**: "a"(1)**

**: "ebx", "ecx"**

**);**

**bool FPU\_bit = (D0 & (1 << 0)) != 0;**

**cout << FPU\_bit << endl;**

**return 0;**

**}**

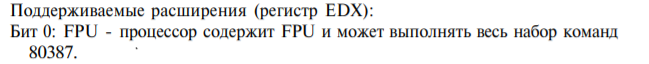
**Вывод:**

1

1

1

**Итог:** AVX, SSE, FPU доступны



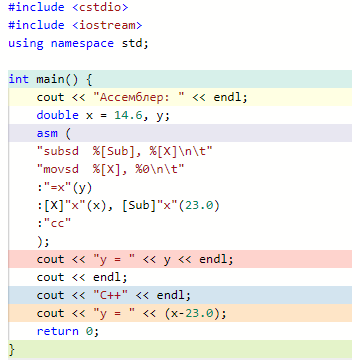


**Задание 4.**

Вычислите для заданного 𝑥 с плавающей запятой двойной точности значение 𝑦 (используйте скалярные AVX-команды 𝑣𝑚𝑜𝑣𝑠𝑑/𝑣𝑎𝑑𝑑𝑠𝑑/𝑣𝑠𝑢𝑏𝑠𝑑, если они доступны, и их SSE-аналоги 𝑚𝑜𝑣𝑠𝑑/𝑎𝑑𝑑𝑠𝑑/𝑠𝑢𝑏𝑠𝑑 в противном случае).

Проверьте расчет, реализовав то же самое на С/C++.

****

****

**SSE**

**#include <cstdio>**

**#include <iostream>**

**using namespace std;**

**int main() {**

**cout << "Ассемблер: " << endl;**

**double x = 14.6, y;**

**asm (**

**"subsd %[Sub], %[X]\n\t"**

**"movsd %[X], %0\n\t"**

**:"=x"(y)**

**:[X]"x"(x), [Sub]"x"(23.0)**

**:"cc"**

**);**

**cout << "y = " << y << endl;**

**cout << endl;**

**cout << "C++" << endl;**

**cout << "y = " << (x-23.0);**

**return 0;**

**}**

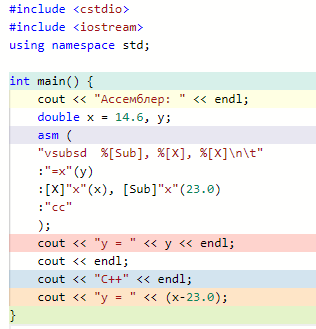
**Ассемблер:**

**y = -8.4**

**C++**

**y = -8.4**

**AVX**

****

**#include <cstdio>**

**#include <iostream>**

**using namespace std;**

**int main() {**

**cout << "Ассемблер: " << endl;**

**double x = 14.6, y;**

**asm (**

**"vsubsd %[Sub], %[X], %[X]\n\t"**

**:"=x"(y)**

**:[X]"x"(x), [Sub]"x"(23.0)**

**:"cc"**

**);**

**cout << "y = " << y << endl;**

**cout << endl;**

**cout << "C++" << endl;**

**cout << "y = " << (x-23.0);**

**}**

Ассемблер:

y = -8.4

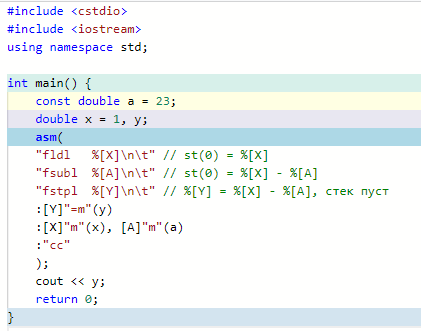
C++

y = -8.4

**Задание 5.**

Вычислите для заданного 𝑥 с плавающей запятой двойной точности значение Л6.з4, используя FPU.

****

****

**#include <cstdio>**

**#include <iostream>**

**using namespace std;**

**int main() {**

**const double a = 23;**

**double x = 1, y;**

**asm(**

**"fldl %[X]\n\t" // st(0) = %[X]**

**"fsubl %[A]\n\t" // st(0) = %[X] - %[A]**

**"fstpl %[Y]\n\t" // %[Y] = %[X] - %[A], стек пуст**

**:[Y]"=m"(y)**

**:[X]"m"(x), [A]"m"(a)**

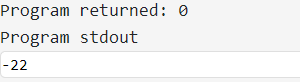
**:"cc"**

**);**

**cout << y;**

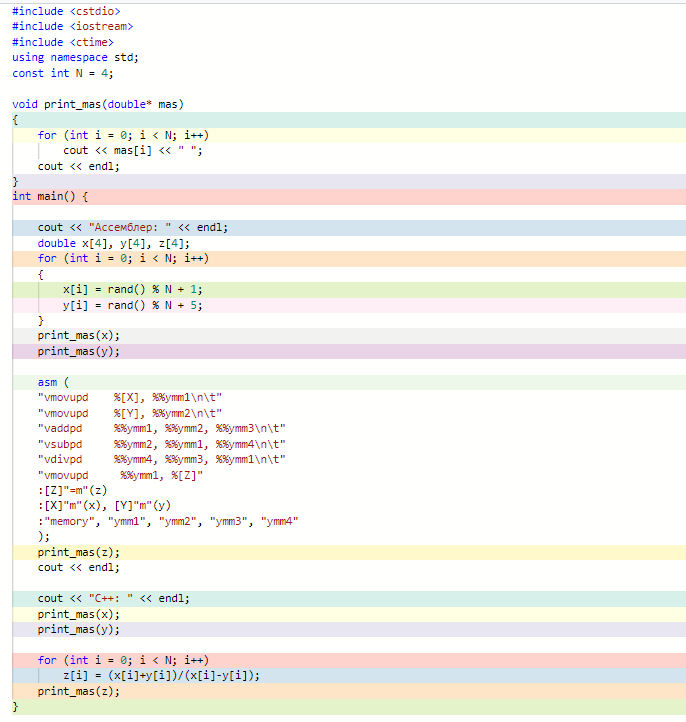
**return 0;**

**}**

****

**Задание 6.**

Рассчитайте, используя векторные команды AVX 𝑣𝑚𝑜𝑣𝑢𝑝𝑑/𝑣𝑎𝑑𝑑𝑝𝑑/𝑣𝑠𝑢𝑏𝑝𝑑/𝑣𝑚𝑢𝑙𝑝𝑑/𝑣𝑑𝑖𝑣𝑝𝑑 и 𝑦𝑚𝑚-регистры (или SSE-аналоги и 𝑥𝑚𝑚) для массивов (𝑥0, ...𝑥3) и (𝑦0, ...𝑦3) из четырёх чисел с плавающей запятой двойной точности (𝑑𝑜𝑢𝑏𝑙𝑒), аналогичный массив (𝑧0, ...𝑧3), где 𝑧𝑖 = (𝑥𝑖 + 𝑦𝑖)/(𝑥𝑖 − 𝑦𝑖). Выделение памяти под 𝑥, 𝑦, 𝑧 и заполнение массивов 𝑥, 𝑦 может быть выполнено на C/C++. Проверьте расчет, реализовав то же самое на С/C++.



#include <cstdio>

#include <iostream>

#include <ctime>

using namespace std;

const int N = 4;

void print\_mas(double\* mas)

{

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

cout << mas[i] << " ";

cout << endl;

}

int main() {

cout << "Ассемблер: " << endl;

double x[4], y[4], z[4];

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

{

x[i] = rand() % N + 1;

y[i] = rand() % N + 5;

}

print\_mas(x);

print\_mas(y);

asm (

"vmovupd %[X], %%ymm1\n\t"

"vmovupd %[Y], %%ymm2\n\t"

"vaddpd %%ymm1, %%ymm2, %%ymm3\n\t"

"vsubpd %%ymm2, %%ymm1, %%ymm4\n\t"

"vdivpd %%ymm4, %%ymm3, %%ymm1\n\t"

"vmovupd %%ymm1, %[Z]"

:[Z]"=m"(z)

:[X]"m"(x), [Y]"m"(y)

:"memory", "ymm1", "ymm2", "ymm3", "ymm4"

);

print\_mas(z);

cout << endl;

cout << "С++: " << endl;

print\_mas(x);

print\_mas(y);

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

z[i] = (x[i]+y[i])/(x[i]-y[i]);

print\_mas(z);

}

Ассемблер:

4 2 2 3

7 8 8 5

-3.66667 -1.66667 -1.66667 -4

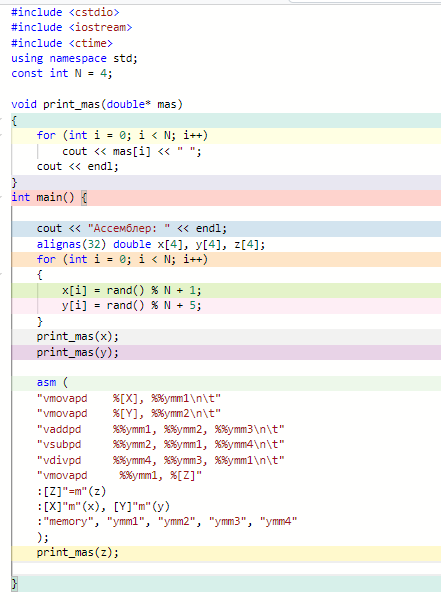
С++:

4 2 2 3

7 8 8 5

-3.66667 -1.66667 -1.66667 -4

**Задание 7.**

****

**#include <cstdio>**

**#include <iostream>**

**#include <ctime>**

**using namespace std;**

**const int N = 4;**

**void print\_mas(double\* mas)**

**{**

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

**cout << mas[i] << " ";**

**cout << endl;**

**}**

**int main() {**

**cout << "Ассемблер: " << endl;**

**alignas(32) double x[4], y[4], z[4];**

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

**{**

**x[i] = rand() % N + 1;**

**y[i] = rand() % N + 5;**

**}**

**print\_mas(x);**

**print\_mas(y);**

**asm (**

**"vmovapd %[X], %%ymm1\n\t"**

**"vmovapd %[Y], %%ymm2\n\t"**

**"vaddpd %%ymm1, %%ymm2, %%ymm3\n\t"**

**"vsubpd %%ymm2, %%ymm1, %%ymm4\n\t"**

**"vdivpd %%ymm4, %%ymm3, %%ymm1\n\t"**

**"vmovapd %%ymm1, %[Z]"**

**:[Z]"=m"(z)**

**:[X]"m"(x), [Y]"m"(y)**

**:"memory", "ymm1", "ymm2", "ymm3", "ymm4"**

**);**

**print\_mas(z);**

**}**

Ассемблер:

4 2 2 3

7 8 8 5

-3.66667 -1.66667 -1.66667 -4