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.

Москва 2022

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

Задание 1.
Вычислите для заданных целых x и y:

2	$z = 5 - (x-1)(y-1)$

```
#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;
}
```

```
#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
```

```
        movl    DWORD PTR [rsp+12], %eax
        subl    $1, %eax
        movl    DWORD PTR [rsp+8], %ebx
        subl    $1, %ebx
        imull   %ebx
        imull   $-1, %eax
        addl    $5, %eax
        movl    %eax, DWORD PTR [rsp+4]
```

Задание 2.

Вычислите для заданного целого x:

8	$z = 16x$

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    int x = 2, z;
    asm (
    "leal   0x0(,%%eax,4), %%eax\n\t"
    :"=a"(z)
    :[X]"a"(x)
    );
    cout << "z = " << z << endl;
    return 0;
}
```

```
#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
```

```
    mov     eax, 2
    leal    0x0(,%eax,4), %eax

    mov     ebx, eax
```

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 << SSE_bit << endl;

    return 0;
}
```

```cpp
#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

Поддерживаемые расширения (регистр EDX):
Бит 0: FPU - процессор содержит FPU и может выполнять весь набор команд 80387.

Бит 25: SSE - процессор поддерживает расширения SSE (Pentium III).

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 C/C++.

2	$y = x - 23$

```
#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;
}
```

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);
}
```

```
#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.

2	$y = x - 23$

```cpp
#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;
}
```

```cpp
#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;
}
```

```
Program returned: 0
Program stdout
-22
```

Задание 6.

Рассчитайте, используя векторные команды AVX *vmovupd*/*vaddpd*/*vsubpd*/*vmulpd*/*vdivpd* и *ymm*-регистры (или SSE-аналоги и *xmm*) для массивов (*x*0, ...*x*3) и (*y*0, ...*y*3) из четырёх чисел с плавающей запятой двойной точности (*double*), аналогичный массив (*z*0, ...*z*3), 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 C/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 << "C++: " << 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);
}
```

```
#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 << "C++: " << 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

C++:
4 2 2 3
7 8 8 5
-3.66667 -1.66667 -1.66667 -4
```

Задание 7.

```cpp
#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);

}
```


```cpp
#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
```