Отчет по 6 лабораторной работе.

Команда: Холодов А.В. Иванов А.А. ПИН-21

Задание Л6.з1. Вычислите для заданных целых x и y:

$$z = x + (x + 1)y - 6$$

Реализация:

```
void exercise_1()
{
    cout<<endl<<"Exercise 1:"<<endl;
    int x,y,z;
    x=5;
    y=5;
    asm(
        "mov %[X], %[Z]\n"
        "inc %[X]\n"
        "imul %[X], %[Y]\n"
        "sub $6, %[Y]\n"
        "add %[Y], %[Z]\n"
        : [Z] "=m" (z)
        : [X] "r" (x), [Y] "r" (y)
        : "cc", "memory"
      );
    cout <<"z = "<< z <<  endl;
}
```

Вывод:

```
Exercise 1:
z = 29
```

Задание Л6.з2. Вычислите для заданного целого x:

```
void exercise_2()
{
    cout<<endl<<"Exercise 2:"<<endl;
    int x=50,z;
    asm(
        "lea 123(%%eax,1), %%eax\n"
        : "=a" (z)
        : "a" (x)
        : "cc"
      );

      cout << "Z = " << z << endl;
}
```

Вывод:

```
Exercise 2:
Z = 173
```

Задание Л6.з3. Проверьте, доступны ли на используемой платформе команды AVX, SSE (SSE1-SSE4.2), FPU.

Реализация:

```
void exercise_3()
{
    cout<<endl<<"Exercise 3:"<<endl;
    int c_reg = 0;
    int d_reg = 0;
    asm(
        "xor %%eax, %%eax\n"
        "mov $1, %%eax\n"
        "cpuid\n"
        :[F] "=&c" (c_reg), [D] "=&d" (d_reg)
        : : "%eax", "%ebx"
    );
    cout<< "c_reg" << bitset<32> (c_reg);
    cout<<endl;
    cout<<"d_reg" <<bitset<32> (d_reg);
    cout<<endl;

    cout<<"Fpu supported(0 bit d_reg)" <<endl;
    cout<<"SSE1 supported(25 bit d_reg)" <<endl;
    cout<<"SSE2 supported(26 bit d_reg)" <<endl;
    cout<<"Avx supported(28 bit c_reg)" <<endl;
    cout<<"SSE3 supported(0 bit c_reg)" <<endl;
    cout<<"SSE4.1 supported(19 bit c_reg)" <<endl;
    cout<<"SSE4.2 supported(20 bit c_reg)" <<endl;
}
```

Вывод:

```
Exercise 3:
c_reg11111111111110100011001000000011
d_reg00011111100010111111101111111111
Fpu supported(0 bit d_reg)
SSE1 supported(25 bit d_reg)
SSE2 supported(26 bit d_reg)
Avx supported(28 bit c_reg)
SSE3 supported(0 bit c_reg)
SSE4.1 supported(19 bit c_reg)
SSE4.2 supported(20 bit c_reg)
```

Задание Л6.з4. Вычислите для заданного x с плавающей запятой двойной точности значение y (таблица Л6.1) (используйте скалярные AVX-команды $vmovsd/vaddsd/vs$

$y = x + 76$

Реализация:

```
void exercise_4()
{
    cout<<endl<<"Exercise 4:"<<endl;
    double x = 0.1;
    double y = 0;
    asm (
      "vaddsd  %[I], %[X], %[X]\n"
      : "=x" (y)
      :[X] "x" (x), [I] "x" (76.0)
      :"cc"
    );
    cout << "y="<< y<<endl;
}
```

Вывод:

```
Exercise 4:
y=76.1
```

Задание Л6.з5. Вычислите для заданного x с плавающей запятой двойной точности значение Л6.з4, используя FPU.

Реализация:

```
void exercise_5()
{
    cout<<endl<<"Exercise 5:"<<endl;
    double x = 0.1;
    double y = 0;
    double add = 76;
    asm(
       "fldl %[X]\n" // st(0) = %[X]
       "faddl %[A]\n" // st(0) = %[X] + %[A]
       "fstpl %[Y]\n" // %[Y] = %[X] + %[A]
        :[Y]"=m"(y)
        :[X]"m"(x), [A]"m"(add)
        :"cc"
    );
    cout << "y="<< y<<endl;
}
```

Вывод:

```
Exercise 5:
y=76.1
```

Задание Л6.з6. Рассчитайте, используя векторные команды AVX $vmovupd/vaddpd/vsubpd/vmulpd/vdivpd$ и ymm-регистры (если они недоступны — 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

$$z_i = (x_i + y_i)/(x_i - y_i).$$

Выделение памяти под x, y, z и заполнение массивов x, y может быть выполнено на C/C++.

Проверьте расчёт, реализовав то же самое на C/C++.

Реализация:

```
void exercise_6()
{
    cout<<endl<<"Exercise 6:"<<endl;
    double x[4] = {1.1, 4.8, 2.6, 5.9};
    double y[4] = {11.3, 1.3, 9.8, 4.4};
    double z[4] = {0, 0, 0, 0};
    asm(
        "vmovupd %[X], %%ymm1\n"
        "vmovupd %[Y], %%ymm2\n"
        "vaddpd %%ymm1, %%ymm2, %%ymm3\n"
        "vsubpd %%ymm2, %%ymm1, %%ymm4\n"
        "vdivpd %%ymm4, %%ymm3, %%ymm3\n"
        "vmovupd %%ymm3, %[Z]\n"
        :[Z] "=m" (z)
        :[X] "m" (x), [Y] "m" (y)
        :"cc", "ymm1", "ymm2", "ymm3", "ymm4", "memory"
    );
    for(int i = 0; i < 4; i++)
        cout << z[i] <<endl;
}
```

Вывод:

```
Exercise 6:
-1.21569
1.74286
-1.72222
6.86667
```

Задание Л6.з7. Реализуйте Л6.з6, используя более быструю команду $vmovapd$ и, соответственно, выравняв x, y, z на 32 байта.

Реализация:

```
void exercise_7()
{
    cout<<endl<<"Exercise 7:"<<endl;
    alignas(32)double x[4] = {1.1, 4.8, 2.6, 5.9};
    alignas(32)double y[4] = {11.3, 1.3, 9.8, 4.4};
    alignas(32)double z[4] = {0, 0, 0, 0};
    asm(
        "vmovapd %[X], %%ymm1\n"
        "vmovapd %[Y], %%ymm2\n"
        "vaddpd %%ymm1, %%ymm2, %%ymm3\n"
        "vsubpd %%ymm2, %%ymm1, %%ymm4\n"
        "vdivpd %%ymm4, %%ymm3, %%ymm3\n"
        "vmovapd %%ymm3, %[Z]\n"
        :[Z] "=m" (z)
        :[X] "m" (x), [Y] "m" (y)
        :"cc", "ymm1", "ymm2", "ymm3", "ymm4", "memory"
    );
    for(int i = 0; i < 4; i++)
        cout << z[i] <<endl;
}
```

Вывод:

```
Exercise 7:
-1.21569
1.74286
-1.72222
6.86667
```

Л6.3. Вопросы

1. Какие вы знаете регистры общего назначения x86 и x86-64?
2. Какие вы знаете команды ассемблера x86?

1. На x86 доступно восемь 32-х битных регистров общего назначения — eax, ebx, ecx, edx, esp, ebp, esi и edi. Архитектура x86-64 имеет 16 целочисленных 64-битных регистров общего назначения (RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP, R8 — R15)

2. Команды передачи данных, Двоичные арифметические команды, Логические команды, Команды работы со строками (последовательностями