# Лабораторная работа 9 (1001 = 9)

Вычисления с плавающей запятой. Команды FPU

**Цель работы**: научиться использовать команды FPU.

## Задание Л9.№1

Вычислите для заданных $x$ и $y$ с плавающей запятой двойной точности выражение из таблицы Л7.1, используя FPU. Результаты заданий Л7.№4 и Л9.№1 напечатайте $print64()$ из Л2. Совпадают ли результаты побитово? Если значения различаются — как вы думаете, какое из них точнее? Достаточно ли они близки друг к другу, чтобы можно было считать их совпадающими в пределах допустимой погрешности?

Вариант 2: $z = 1 - 5/x - y^2/7$

```
Задание №1
========================================================================
SystemInfo
========================================================================
ОС: Linux
Архитектура процессора: x86_64 (64-бит)
Compiler: GCC
Version: 13.2.1
========================================================================
z = 1 - 5/2.000 - 3.000^2/7 =
fpu: C006492492492492 1100000000000110010010010010010010010010010010010010010010010010 13836827326564344978 -4609916747145206638 -0x1.6492492492492p+1 -2.785714e+00 -2.79
avx: C006492492492492 1100000000000110010010010010010010010010010010010010010010010010 13836827326564344978 -4609916747145206638 -0x1.6492492492492p+1 -2.785714e+00 -2.79
========================================================================
```

**Рис. 1:** Результат вычисления с использованием AVX и FPU

Листинг:

**Файл task9_1.c:**

```c
void run_task9_1()
{
    printf("\nЗадание №1\n");
    printf("================================================================================");
    printSystemInfo();
    printf("================================================================================\n");

    double x = 2.0, y = 3.0;

    double z_fpu = FPU_compute(x, y);
    double z_avx = avx_compute(x, y);

    printf("z = 1 - 5/%.3f - %.3f^2/7 = \n", x, y);
    printf("fpu: "); print64(&z_fpu);
    printf("avx: "); print64(&z_avx);

    printf("================================================================================\n");
}

double FPU_compute(double x, double y)
{
    double z;
    const double five = 5.0,  seven = 7.0, one = 1.0;

    __asm__ volatile (
        "fldl %2\n\t"               // Загружаем y в st0
        "fmul %%st(0), %%st(0)\n\t" // st0 = y²
        "fldl %5\n\t"               // st0 = 7, st1 = y²
        "fdivrp \n\t"               // st0 = y²/7                          ST(1)/ST(0)

        "fldl %1\n\t"               // st0 = x, st1 = y²/7
        "fldl %3\n\t"               // st0 = 5, st1 = x, st2 = y²/7
        "fdivp \n\t"                // st0 = 5/x, st1 = y²/7               ST(0)/ST(1)

        "fldl %4\n\t"               // st0 = 1, st1 = 5/x, st2 = y²/7
        "fsubp \n\t"                // st0 = (1-5/x) - y²/7                ST(0) – ST(1)
        "fsubp \n\t"                // st0 = (1 - 5/x - y²/7)              ST(0) – ST(1)
        "fstpl %0\n\t"              // Сохраняем результат в z
        : "=m"(z)
        : "m"(x), "m"(y), "m"(five), "m"(one), "m"(seven)
        : "st", "st(1)", "st(2)"
    );
    return z;
}
```

## Задание Л9.№2

Вычислите для заданных $x$ и $y$ с плавающей запятой двойной точности выражение из таблицы Л7.2, используя FPU и не используя функций libm.

Результаты заданий Л7.№5 и Л9.№2 напечатайте $print64()$ из Л2 и сравните аналогично Л9.№1.

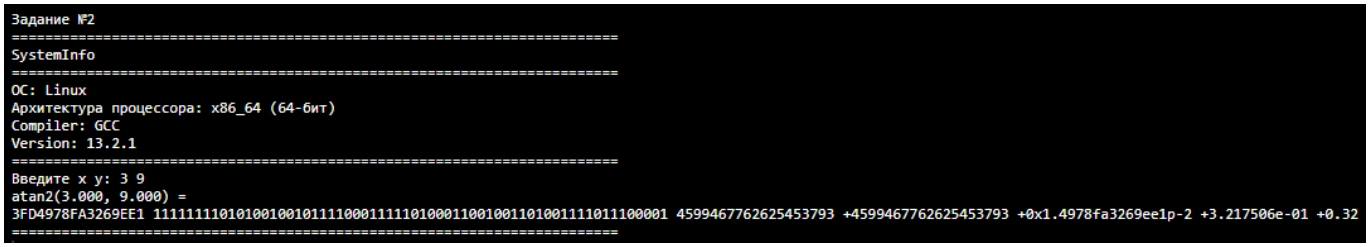


**Рис. 2:** результат выполнения

Листинг:

**Файл task9_2.c:**

```c
double fpu_atan2(double x, double y);

void run_task9_2()
{
    printf("\nЗадание №2\n");
    printf("================================================================================");
    printSystemInfo();
    printf("================================================================================\n");

    double x, y;
    printf("Введите x y: ");
    if (scanf("%lf %lf", &x, &y) != 2) {
        printf("Ошибка ввода\n");
        return;
    }

    double z_fpu = fpu_atan2(x, y);
    printf("atan2(%.3f, %.3f) = \n", x, y);
    print64(&z_fpu);

    printf("================================================================================\n");
}

double fpu_atan2(double x, double y)
{
    double z;
    __asm__ volatile (
        "fldl %1\n\t"       // st0 = x
        "fldl %2\n\t"       // st0 = y, st1 = x
        "fpatan\n\t"        // st0 = atan2(x, y)
        "fstpl %0\n\t"
        : "=m"(z)
        : "m"(x), "m"(y)
        : "st", "st(1)", "st(2)"
    );
    return z;
}
```

## Задание Л9.№3

Вычислите для заданных $x$ и $y$ с плавающей запятой двойной точности выражение из таблицы Л9.1, используя FPU.

Вариант 2: $y \cdot \log2\ (x + 1)$.

```
Задание №3
========================================================================
SystemInfo
========================================================================
ОС: Linux
Архитектура процессора: x86_64 (64-бит)
Compiler: GCC
Version: 13.2.1
========================================================================
Введите x y: 3 9
z = 9.000 - log2(3.000 - 1) =
4032000000000000 1000000001100100000000000000000000000000000000000000000000000000 4625759767262920704 +4625759767262920704 +0x1.2p+4 +1.800000e+01 +18.00
========================================================================
```

**Рис. 3:** результат вычисления fyl2xp1

Листинг:

**Файл task9_3.c:**

```c
double FPU_compute3(double x, double y);

void run_task9_3()
{
    printf("\nЗадание №3\n");
    printf("========================================================================");
    printSystemInfo();
    printf("========================================================================\n");

    double x, y;
    printf("Введите x y: ");
    if (scanf("%lf %lf", &x, &y) != 2) {
        printf("Ошибка ввода\n");
        return;
    }

    double z_fpu = FPU_compute3(x, y);
    printf("z = %.3f * log2(%.3f - 1) = \n", y, x);
    print64(&z_fpu);

    printf("========================================================================\n");
}

double FPU_compute3(double x, double y) {
    double z;
    __asm__ volatile (
        "fldl %2\n\t"           // st0 = y
        "fldl %1\n\t"           // st0 = x, st1 = y
            "fyl2xp1\n\t"           // st1 * log2(st0) -> st0
        "fstpl %0\n\t"
        : "=m"(z)
        : "m"(x), "m"(y)
        : "st"
    );
    return z;
}

```

## Задание Л9.№4.

Реализуйте Л5.№2 для $x$ с плавающей запятой (таблица Л9.2), используя FPU-команды сравнения $f[u]comi[p]$.

Вариант 2: Двойной точности ($double$)

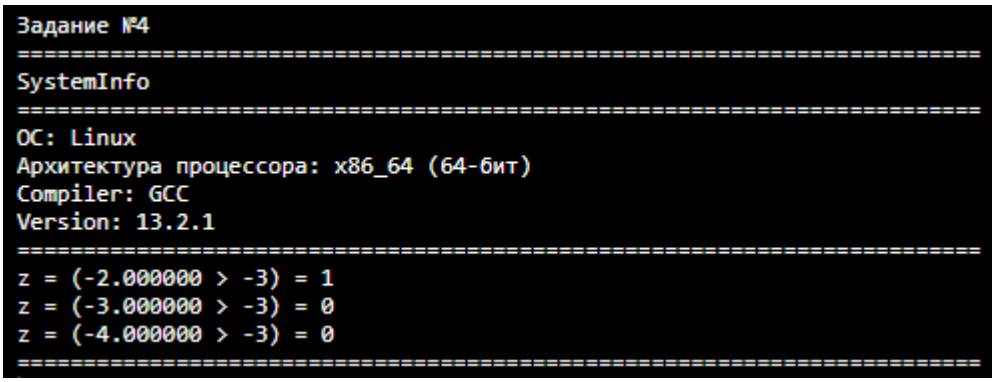


**Рис. 4:** выполнение сравнения при помощи fcomip

Листинг:

**Файл task9_4.c:**

```c
int FPU_compute4(double x);

void run_task9_4()
{
    printf("\nЗадание №4\n");
    printf("================================================================================");
    printSystemInfo();
    printf("================================================================================\n");

    double x1 = -2.0, x2 = -3.0, x3 = -4.0;
    int z;

    z = FPU_compute4(x1);
    printf("z = (%f > -3) = %i\n", x1, z);

    z = FPU_compute4(x2);
    printf("z = (%f > -3) = %i\n", x2, z);

    z = FPU_compute4(x3);
    printf("z = (%f > -3) = %i\n", x3, z);

    printf("================================================================================\n");
}

int FPU_compute4(double x) {
    int result;
    const double compare_v = -3.0;

    __asm__ volatile (
        "fldl %2\n\t"                   // st0 = -3
        "fldl %1\n\t"                   // st0 = x, st1 = -3
        "fcomip \n\t"                   // сравнить st0(-3) и st1(x), удалить st0
        "fstp %%st(0)\n\t"              // очистить стек FPU
        "seta %b0\n\t"                  // если x > -3, seta = 1, иначе 0
        : "=r" (result)
        : "m"(x), "m"(compare_v)
        : "cc", "st", "st(1)"
    );
    return result;
}
```