Лабораторная работа №1
Представление чисел.

1. Работа была выполнена с использованием среды разработки VSCode, язык программирования C++, компилятор g++ (Debian 12.2.0-14) 12.2.0.

2.

input	**output**
8.8 2 0x9c9f	-99.378
16.16 2 0x6f7600 + 0x1713600	134.672
f 2 0x414587dd * 0x42ebf110	0x1.6c1b74p+10
f 2 0x414587dd + 0x42ebf110	0x1.04a20cp+7
h 2 0x4145 * 0x42eb	0x1.23cp+3
h 2 0x8000 + 0x0	0x0.000p+0
f 2 0x0	0x0.000000p+0
f 2 0x7f800000	inf
f 2 0xff800000	-inf
f 2 0x7fc00000	nan
f 2 0x1 / 0x0	inf
f 2 0xff800000 / 0x7f800000	nan

3. Если ввод не корректный или происходит деление на ноль чисел с фиксированной точкой, обрабатывается ошибка invalid_argument и программа завершается с не нулевым кодом ошибки.

Для удобства были созданы отдельные классы для каждого типа данных.
Введённые числа приводятся к типу uint64_t при помощи метода strtoull и дальше передаются в конструктор класса, в зависимости от введённого типа числа.

FixedPoint:
Числа хранятся как беззнаковый тип целых чисел.

1. Для сложения и вычитания данных чисел достаточно произвести соответствующие операции с целыми числами и отбросить лишние биты слева.

2. Для умножения необходимо перемножить модули целых чисел, округлить результат, отбросить лишние биты и, если результат должен быть отрицательным,

найти противоположное число. Число после умножения надо округлить, если в первых B битах справа есть не нулевые биты и число положительное.

3. Деление: делим модули чисел, сначала находится целая часть деления чисел друг на друга, дальше при помощи деления в столбик находятся следующие B бит – дробная часть числа. Если после получения дробной части делимое не равно нулю и число положительное, то число надо округлить. Потом лишние биты числа отбрасываются и, если результат должен быть отрицательным, найти противоположное число.

4. Округление: Число с фиксированной точкой представляется в дополнении до двух, поэтому все не нулевые биты кроме последнего надо прибавлять, но когда мы отбрасываем не нулевые биты, число становится меньше, то есть нам надо прибавить к числу 1, чтобы оно стало больше и это не зависит от знака числа.

5. Вывод: Для вывода берётся модуль числа. Дальше делением в столбик находятся необходимые цифры. Если делимое осталось не нулевым, число округляется.

Single/Half presision:
Введённое число раскладывается на знак, экспоненту и мантиссу.
Если число денормализованное, экспонента заменяется на минимальное допустимое значение(single: -126, half: -14). Если чосло нормальное, то к мантиссе добавляется скрытая единица.
Обрабатываются особые случаи 0, ±inf, nan.

1. Умножение: проверяем особые случаи, если это обычные числа, то перемножаем мантиссы, складываем экспоненты и находим xor двух знаков. Дальше передаем полученное значение в функцию rearrange.

2. Деление: проверяем особые случаи. Делим мантиссы столбиком, пока не получим 25 цифр. Если делимое не 0 и результат деления – положительное число, первый бит меняем на единицу. Полученное значение передаем в функцию rearrange.

3. Сложение: обрабатываем особые случаи. Находим наибольшее число по модулю. Для сложения надо меньшее число привести к экспоненте большего числа. Находим их сумму, и проверяем отбросили ли мы ненулевые биты. Если да, то возможны два случая:

а) Оба слагаемых положительные, тогда мы отбросили часть суммы и получили ответ, меньше чем надо и мы заменим первый бит результата на 1.

б) Второй числа имеют разные знаки, тогда мантиссу уменьшаем на один и если сумма положительная, первый бит меняем на 1. Объяснение: пусть A – большее число по модулю, b – меньшее. Есть два варианта:

1. $A - b = S - d$, где S – полученный нами результат, d это то, что мы отбросили($0 < d < 1$), тогда $S - d = (S - 1) + (1 - d)$ – уменьшив S на 1, мы вернёмся к случаю а).

2. $b - A = -S + d = -(S - 1) - (1 - d)$ – уменьшив S на 1 мы получим отрицательное число, которое ближе к нулю, а $-(1 - d)$ можно игнорировать.

4. Вычитание: меняем знак второго числа и возвращаемся к сложению.

5. Rearrange: Находим первую единицу справа и ставим её на 24 позицию(11 для half). Если число положительное и мы отбросили не нулевые биты(поэтомы мы ставили единицу на первую позицию) его надо будет округлить.
Проверяем число на 0, ±inf. Если оно денормализованное, то приводим экспоненту к минимально допустимому по IEEE754 и если мы ранее или сейчас отбросили ненулевые биты, мы округляем и снова применяем rearrange, так как после округления могло получится нормальное число. Если число нормальное проверяем отбрасывали ли мы ранее биты и если да, то округляем и снова запускаем rearrange, так как могли попасть в бесконечность.

6. Вывод: обрабатываем особые случаи, приводим первую единицу на 25 позицию (13 для half) и все числа справа выводим в шестнадцетиричном представлении.