Миниотчёт по дабораторной работе № 1

(Тип округления - к нулю)

Инструментарий:

Компилятор: gcc (GCC) 12.2.1

Язык C11

1. Fixed point precision

- Все числа хранятся в дополнении до двух в формате “A.B”. Хранить будем просто как набор битов в типе unsigned int, так как $A + B \leq 32$
- Следовательно

$$-x_{\text{bits}} = \sim x_{\text{bits}} + 1$$

*Здесь и далее x_{bits} – число (unsigned) представляющее наше число x побитово (то есть так, как мы его храним)

- Выводить надо с тремя знаками после запятой. Сначала просто выведем целую часть числа. (Если исходное число отрицательное, то сначала приведем его к виду положительного и выведем знак минус вначале). Затем у нас останется дробная часть $\text{frac} = \frac{\text{frac}_{\text{bits}}}{2^b}$, которую нам необходимо представиь в десятичном виде. Тогда наши три цифры после запятой (z) будут иметь вид (так как округление к нулю):
$z = \left\lfloor \frac{\text{frac}_{\text{bits}}}{2^b} \cdot 10^{b-3} \right\rfloor = \left\lfloor \frac{\text{frac}_{\text{bits}}}{2^{b-3}} \cdot 125 \right\rfloor = (\text{frac}_{\text{bits}} \cdot 125) \gg (b-3)$
- Сложение и вычитание примитивны, достаточно сложить x_{bits} и y_{bits}.
- С умножением сложнее. Вначале поймем знак нашего выражения и возьмем оба числа по модулю. Тогда результатом умножения будет $\frac{x_{\text{bits}} \cdot y_{\text{bits}}}{2^b}$ с нужным округлением и нужным знаком. Формула очевидна если расписать столбиком то, как производится умножение. Делим на 2^b так как получается $2b$ знаков после запятой, а нам нужно оставить только b знаков. После приведения к нужному знаку, который мы определили до начала умножения, число могло в результате переполнения опять стать

отрицательным. Для этого, если при делении на 2^b оставался какой-то остаток, то сделаем прибавление 1 к res_{bits}, так как числа хранятся в коде дополнения до двух, потому что 0 будет больше.

- В делении $\frac{x}{y} = \frac{x_{\text{bits}}}{y_{\text{bits}}}$ Чтобы получить b знаков после запятой умножим x на 2^b, тогда ответом будет $\left\lfloor \frac{x_{\text{bits}} \cdot 2^b}{y_{\text{bits}}} \right\rfloor$. Сначала аналогично определяем знак выражения и берём по модулю.

2. Single point precision

- Хранятся в формате 1.8.23 бита под знак, экспоненту и мантиссу, храним в типе unsigned int
- Экспонента хранится в формате сдвига на -127
- В денормализованно числе $\text{exp}_{\text{bits}} = 0$ и считается что $\text{exp} = -126$. Числам соответствуют числа вида $0.m \cdot 2^{-126}$
- Нормализованное число представляется в виде $1.m * 2^{\text{exp}}$
- Отдельные значения : +inf, -inf, nan, +0, -0
- Вывод. если число - это одно из отдельных значений то выведем его. Если число денормализованное, то сдвигаем мантиссу на нужное число битов, чтобы получить 1 в нужном бите и прибавляем к экспоненте число на которое сдвинули. Чтобы вывести мантиссу в шестнадцатеричном виде делаем ее сдвиг влево на 1, чтобы количество битов стало кратно 4.
- Сложение. Сделаем так, чтобы фукнции передавалось два положительных числа, если это не так то вызовем вычитание двух положительных или вызовемся от противоположных аргументов. В случае сложения двух денормализованных чисел ответом будет $a_{\text{bits}} + b_{\text{bits}}$. Чтобы сложить приводим число с меньшей экспонентой к экспоненте большего числа. Так как округление к нулю, то можно откинуть лишние биты. Работаем только с мантиссами. Ставим в нужной позиции (23 бит) единицу если число нормализованное (делаем это до сдвига на разницу экспонент). Затем складываем как числа с фиксированной точкой. Приводим к нормализованному виду если можем. А

- Вычитание. Всё аналогично сложению. Но если $a < b$ меняем их местами и берем ответ со знаком минус.
- Умножение. Умножаем числа как числа с фиксированной точкой, экспоненты складываем.
- Деление. Делим так же как числа с фиксированной точкой, экспоненты вычитаем.

3. Half point precision

- Хранятся в формате 1.5.10 под знак, экспоненту и мантиссу, храним в типе unsigned short
- Все операции аналогичны *single point precision*