Первая лабораторная работа по Архитектуре ЭВМ

Сайфуллин Искандар M3136

10 октября 2023

Использовался **Python 3.11.5**
Весь код находится в *main.py*. Реализованы 2 класса: `FloatingPointNumber` и `FixedPointNumber`

1 Ввод

В функции `check_input` проверяется:

1. Правильность формата числа
2. Округление (Поддерживается только округление к $-\infty$)
3. Правильность операции
4. Формат чисek. Если слишком большое или не начинается с `0x`, то программа завершается

Если всё хорошо, то программа продолжает работу, иначе выводит все ошибки, которые были найдены в `stderr`.

2 Фиксированная точка

Число храним как обычное целое число без знака, а также знак числа.

$$N = \frac{\hat{N}}{2^B}$$

Где $\hat{N}$ — хранимое число, а N — само число

2.1 Вывод

Для дробной части берём первые B битов. Для целой берём следующие $A - 1$ битов целого числа. Целую часть просто переводим в десятичную систему и отнимаем 2^A, если самый старший бит равен единице.

Дробную часть умножаем на 1000 и сдвигаем на B бит направо, таким образом получая 3 цифры после запятой в десятичной записи.

$$fraction_{10} = \left\lfloor \frac{fraction_2 * 10^3}{2^B} \right\rfloor$$

Если число отрицательное, то вычитаем из 1000 $fraction_{10}$, а также прибавляем единицу к целой части, если $fraction_{10} \neq 0$, так как в таком мы позаимствовали единицу из старших разрядов.

Округление к $-\infty$ происходит при положительных автоматически (так как мы просто отбрасываем все младшие разряды) и при вычитании из 1000 в случае отрицательных

2.2 Сложение и вычитаение

В случае сложения банально складываем два числа по модулю 2^{A+B}, так как числа у хранятся в беззнаковом виде. В случае вычитания инвертируем все биты, прибавляем единицу и также складываем числа по тому же модулю.

2.3 Умножение и деление

Для умножения и деления нам необходимо перевести числа в числа со знаком, чтобы не рассматривать разные случаи с переменой знака.

Умножение:

$$Z = X * Y = \frac{\hat{X}}{2^B} * \frac{\hat{Y}}{2^B} = \frac{\hat{X} * \hat{Y}}{2^{2B}}$$

Следоватеьно, чтобы получить $\hat{Z}$, нам необходимо сдвинуть полученное произведение на B вправо.

Деление:

$$Z = X/Y = \frac{\hat{X}}{2^B} / \frac{\hat{Y}}{2^B} = \frac{\hat{X}}{\hat{Y}}$$

Значит, необходимо сдвинуть частное на B влево.

Замечание: При делении на 0 программа выводит `error` и завершает работу

В конце обеих операций $\hat{Z}$ переводится обратно в запись дополнения до 2.

3 Плавающая точка

3.1 Хранение

В класс хранится:

- мантиссу
- экспоненту и *bias*
- знак числа
- все специальные значения и нули

3.2 Вывод

Сначала проверяется, являетлся ли число нулём, nan'ом или бесконечносью.

Далее выполняется проверка на то, является ли число денормализованным. Если да, то мантисса сдвигается налево на нужное количесвто битов.

После этого смотрим на знак числа, переводим мантиссу в 16-ричную систему счисления и отнимаем *bias* из экспоненты и выводим в нужном формате.

3.3 Сложения и вычитание

Сначала проверяем числа на специальные значения и нули.

Далее приводим числа к одной екпоненте, сдвигаем мантиссы, складываем или вычитаем мантиссы. Вычисляем знак итогого числа и сдвигаем мантиссу влево или вправо, прибавляя или отнимая единицу из экспоненты. В этом процессе может выяснится, что полученное число равно 0, бесконечности или является денормализованным. В конце возварщаем тот же тип.

3.4 Умножение и деление

При умножении и делении тоже проверяем сначала специальные случаи.

Далее знак числа определяется через *xor*, а экспоненты складываются или вычитаются

В случае умножения мантиссы просто перемножаются, а далее обрезаются (с округлением) Также, если после умножения размер мантиссы занимает ровно вдвое больше битов, то прибавляем единицу к экспоненте.

В случае деления сначала сдвигаем мантиссу делимого влево, чтобы вычисления были точнее, а далее также двигаем мантиссу.

В конце обращаем внимание на то, стала ли число бесконечностью или нулём

4 Тесты

Все предоставленные тесты программа проходит верно.

Test	Result	Expected
`8.8 3 0x9c9f`	`-99.379`	`-99.379`
`16.16 3 0x6f7600 + 0x173600`	`134.671`	`134.671`
`f 3 0x414587dd * 0x42ebf110`	`0x1.6c1b72p+10`	`0x1.6c1b72p+10`
`f 3 0x414587dd + 0x42ebf110`	`0x1.04a20ap+7`	`0x1.04a20ap+7`
`h 3 0x4145 * 0x42eb`	`0x1.238p+3`	`0x1.238p+3`
`h 3 0x8000 + 0x0`	`-0x0.000p+0`	`-0x0.000p+0`
`f 3 0x0`	`0x0.000000p+0`	`0x0.000000p+0`
`f 3 0x7f800000`	`inf`	`inf`
`f 3 0xff800000`	`-inf`	`-inf`
`f 3 0x7fc00000`	`nan`	`nan`
`f 3 0x1 / 0x0`	`inf`	`inf`
`f 3 0xff800000 / 0x7f800000`	`nan`	`nan`