Лабораторная работа №1	Б06	2023
Представление чисел	Прозорова Варвара Владиславовна	

Инструментарий и требования к работе: python 3.11.5

Ссылка на репозиторий:

https://github.com/skkv-mkn/mkn-comp-arch-2023-fixed-floating-stepashka-21

Результат работы на тестовых данных:

"h 1 0x3400 + 0x8001" "0x1.000p-2"
"h 2 0x1800 * 0x0200" "0x1.000p-24"
"f 1 0x414587dd / 0x42ebf110" "0x1.aca5aep-4"
"16.16 2 0x6f7600 + 0x173600" "134.672"
"27.5 0 0xfffffffe" "-0.062"
"3.2 1 0x1f - 0x1" " -0.500"
"3.2 2 0x1 * 0x1" "0.250"
"3.2 1 0x1c / 0x4" "-1.000"
"f 0 0xff800000 / 0x7f800000" "nan"

Как работает код?

Для начала посмотрим на числа с плавающей точкой.

На вход подаются числа, записанные в 16-ричной побитовой форме с префиксом ‘0x’, поэтому первое, что надо сделать – перевести каждый знак в записи в двоичный код. Это делает функция perevod1 (каждую цифру кодирует в четыре бита). Дальше функции hr и fr (в зависимости от формата) разбивают получившуюся запись на три части – знак, экспоненту, мантиссу (числа хранятся в виде списка). Переводом числа из двоичного вида обратно в шестнадцатиричную форму занимается функция perevod2, которая получает на вход само число уже нужной длины (увеличенной в 4 раза, конечно) и каждые четыре бита заменяет на число в шестнадцатиричной записи).

Операции:

- round – округление мантиссы – получает на вход знак числа, его мантиссу, тип округления и длину мантиссы, которая должна получиться (зависит от формата).

  Сам процесс округления. Для положительного числа (для отрицательного типы 2 и 3 поменяются):

  0 и 3: просто обрубаем число

  1: смотрим на следующий бит после, дальше либо обрубаем, либо прибавляем 1

  2: на наличие ‘1’ во всем хвосте мантиссы, дальше либо обрубаем, либо прибавляем 1

  Пометки ‘k’ и ‘s’ говорят о том, что произошло переполнение (то есть было, например, число 1111, добавили 1 и получилось 1|0000). Тогда итоговая экспонента должна будет измениться на ± 1.

  В начале каждой следующей операции рассматриваются спец случаи с ‘nan’, ‘±inf’, ‘±0x0.0..’.

- star – умножение – получает на вход два числа, их формат и тип округления. Как работает? Приводим мантиссу к нормальному виду (дописываем в начале ‘1’, а если число денормализованное, то сдвигаем мантиссу до первой единицы за границей длины мантиссы и в счетчик k, который потом уйдет в экспоненту, добавляем размер сдвига) перемножаем как int’ы, округляем. Экспонента получается из экспонент данных чисел + возможный сдвиг (k). Операция округления – `star('0x3F800000', число, f, тип округления)` или `star('0x3C00', число, h, тип округления)`, где первые числа – закодированные единиц, то есть происходит обычное умножение на 1.
- slash – деление – работает аналогично star

- plus – сумма – сначала вычисляется знак итогового числа (для этого все числа сдвигаются на нужное число разрядов и переводятся в int’ы). Знак (sign) запоминается и будет использоваться только для вывода ответа. Далее денормализованные числа приводятся к нормальному виду с изменением экспоненты (или счетчика k). Затем производится сложение мантисс, при этом если у чисел разные знаки, то сумма = abs(разность), иначе сумма = abs(сумма). После всего этого по классике происходит округление и вывод ответа.
- minus – разность – plus для чисел a и -b. То есть вызывается функция plus, где у аргумента b сменился первый бит (0 на 1 и наоборот)

(это не конец отчёта..)

Теперь посмотрим на числа с фиксированной точкой.

На вход подаются числа, записанные в 16-ричной побитовой форме с префиксом ‘0x’, поэтому первое, что надо сделать – перевести каждый знак в записи в двоичный код. Это делает функция perevod1, как в числах с плавающей точкой, а функция fix корректирует длину, получая на вход числа a и b из заданного на входе формата, то есть докидывает нужное число в начало записи. В начале каждой операции числа переводятся из кода с дополнением до 2 в нормальный вид.

Операции:

- plus_fix – сложение – получает на вход числа, формат (a и b) и тип округления. Работает так: вычисляется сумма int’ов, берется такой срез числа, чтобы был формат a.b, и потом округляется.
- minus_fix – вычитание – та же самая plus_fix, только второе число делается противоположным по знаку (в формуле int будем с другим знаком).
- star_fix – умножение – все операции производятся над положительными числами, но если результат должен быть отрицательным (sign == -1), то минус дописывается в ответ. Как работает? Изначально sign = 1, потом просто смотрим на первый бит каждого числа: 1 – отрицательное и sign *= 1. После умножения int’ов переводим числа в двоичный вид (длина a + b). Но при перемножении положительных чисел может возникнуть ситуация, когда число получилось больше/меньше максимума/минимума, которое может храниться, тогда еще нужно перевести в нужный вид (берем по модулю). Также если по длине получилось слишком короткое число, то нужно докинуть нулей. Производим округление.

- slash_fix – деление – аналогично умножению, только к делимому дописывается с краю очень много нулей, чтобы сделать корректное округление.
- Print – округление – на вход подаются число, формат (a и b), тип округления, символ r – округление (1) или операция (!= 1) (нужно для определения того, какое число подано), знак, если операция равен -1 или 1 соответственно. При смене знака в округлении меняется тип 3 на 2 и 2 на 3, что заменяется // 6. Округление делается в int'ах с докидыванием кучи нулей, чтобы все сделать точнее.

  В переменную tail будет записываться десятичная дробь и тут может возникнуть проблема, что дробь начинается с нуля, для этого нужно отдельно проверять биты в числе b.

  Целую часть храним int'ом, дробную из двоичной переводим в десятичную, округляем по правилам.

<u>Что изменилось в работе?</u>

- Дополнила отчет.
- Исправила ошибки при считывании входных данных (до этого код не считывал тип округления и не всегда правильно понимал формат числа с плавающей точкой).
- Добавила спец случай: вывод 0x0.00.., когда экспонента меньше минимальной, вывод ±0x1.000p-24 или ±0x1.000000p-149 в случае особого округления.
- Исправила некоторые проблемы с денормализованными числами во всех операциях.
- Рассмотрела случай, когда у tail в функции Print округления фиксированной точки может быть 0 в начале.

Это основные моменты. Были еще изменения (примерно везде), но они незначительные.