Assignment 1 : Binary Encoding

Coding Documentation

Due Date: September 12, 2025

Introduction

The goal of this assignment is to write a program that converts a decimal number string into IEEE-754 64-bit binary. To complete this task, I wrote a Python script that takes a decimal number as input and outputs its binary representation in IEEE-754 64-bit format.

Cases Handled

The following cases are considered:

- Invalid inputs (non-numeric strings)
- Infinity (+ / -)
- NaN
- Zero (+ / -)
- Subnormal numbers (+ / -)
- Decimals that are too large to be represented in IEEE-754 64-bit format
- General case for valid decimal numbers within the representable range.

Algorithm

The algorithm follows these general steps:

1. Parse the input string and validate that it is a decimal number.
2. Handle special cases (e.g., NaN, Infinity, Zero).
3. Check the magnitude of the number to determine if it is subnormal or too large for IEEE-754 64-bit representation.
4. Convert the valid decimal number to its binary representation.
   (a) Determine the sign bit (0 for positive, 1 for negative).

(b) Separate the number into its integer and fractional parts.

(c) Convert both parts to binary, rounding/appending to keep 53 significant bits for the mantissa (pre-normalization) as needed.

(d) Find the exponent.

(e) Calculate the biased exponent and its binary representation. Then prepending with leading zeros to ensure it is 11 bits long if needed.

(f) Combine the integer and fractional binary parts to form the mantissa.

(g) Normalize the mantissa to 52 bits.

(h) Combine the sign, exponent, and mantissa to form the final binary representation.

Note: Extra exception handling occurs in steps 4b and 4d to ensure that conversion is possible within the default precision (28 significant digits) of the Decimal object. Due to this limitation, the actual range of representable numbers in this program is roughly 1e-14 to 1e27.

Function Documentation

Main Function

`decimal_to_binary(num_string)`

Description:

This function takes a decimal (string) and returns the number in IEEE 754 64-bit binary form (string). The returned string has spaces to indicate the sign, exponent, and fraction.

Parameters:

`num_string` : `(string)` The decimal provided from user input.

Returns:

`string` : `num_string` converted to IEEE 754 64-bit binary. Formatted as "sign exponent fraction".

Raises:

`TypeError`: If the input is not a valid decimal number.

`ValueError`: If the input is too large for IEEE 754 64-bit conversion, or if the Decimal object cannot represent the input with its default precision.

Helper Functions

`integer_to_binary(integer)`

Description:

This function takes an integer and returns its binary representation as a list of chars. Returns an empty list if the integer entered is any form of 0.

Parameters:

`integer` : `(int` or `float)` The integer to be converted into binary.

Returns:

`list` : The binary representation of the integer as an ordered list. Each index contains a '0' or '1'.

`fraction_to_binary(fraction, integer_binary)`

Description:

This function takes the fractional portion of a number (as a Decimal object) and returns a tuple containing:

1. The binary representation of the fraction as a list of chars.
2. The (possibly rounded) binary representation of the integer part.

The length of the binary fraction list is determined by the number of bits remaining in the mantissa after accounting for the integer part.

Parameters:

`fraction` : `(Decimal)` The fractional portion of the number to be converted into binary.

`integer_binary` : `(list of string)` The binary representation of the integer portion as a list of '0' and '1' chars. Used to calculate the remaining bits in the mantissa before normalization.

Returns:

`tuple (list of string, list of string)` :

- `tuple[0]: list of string`

  Contains the binary representation of the fraction as an ordered list of '0' and '1' chars.

- `tuple[1]: list of string`

  Contains either a rounded version of `integer_binary` (if rounding occurs) or a copy of the original `integer_binary`.

`round_up(binary)`

Description:

This function is a helper for `fraction_to_binary()`.

It rounds up the given binary number (provided as a list of '0' and '1' chars), starting from the least significant bit, *without prepending a new bit*.

It returns a tuple containing:

1. The rounded binary number as a list of '0' and '1' chars.
2. A boolean indicating whether a '1' needs to be appended to the beginning from rounding.

Parameters:

`binary` : `(list of string)` The binary representation of a number as an ordered list of '0' and '1' chars.

Returns:

`tuple (list of string, bool) :`

- `list of string`: The first element is the rounded binary number as a list of '0' and '1' chars.
- `bool`: The second element is a boolean indicating if a '1' needs to be appended to the beginning due to rounding.
