4190.308: Computer Architecture (Fall 2018)

Project #2: TinyFP (8-bit floating point) Representation

Due: October 28th (Sunday), 11:59PM

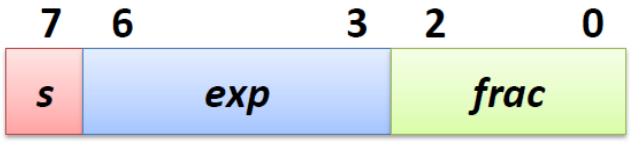
1. Introduction

The purpose of this project is to get familiar with the floating-point representation by implementing a simplified 8-bit floating-point representation.

2. Problem specification

2.1 Overview

`tinyfp` is a simplified 8-bit floating point representation which follows the IEEE 754 standard for floating-point arithmetic. The overall structure of the **`tinyfp`** representation is shown below. The MSB (Most Significant Bit) is used as a sign bit (***s***). The next four bits are used for exponents (***exp***) with a bias value of 7. The last three bits are used for the fractional part (***frac***).



In C, the new type **`tinyfp`** is defined as follows.

```
typedef unsigned char tinyfp;
```

Your task is to implement the following four C functions that convert **`int`** or **`float`** type values to the **`tinyfp`** format and vice versa.

```
tinyfp int2tinyfp(int x);
int tinyfp2int(tinyfp x);
tinyfp float2tinyfp(float x);
float tinyfp2float(tinyfp x);
```

2.2 Implementation details

2.2.1. `int2tinyfp()`

- Integer zero (0) should be converted to plus zero (+0.0) in **`tinyfp`**.
- An integer value that exceeds the range of the **`tinyfp`** representation should be converted to the infinity in **`tinyfp`** (+∞ or -∞ depending on the sign).
- If necessary, use the **round-to-even** mode.

2.2.2. `tinyfp2int()`

- Drop the fractional part when you convert values in the **`tinyfp`** format to integers. (e.g., the value 1.5 in **`tinyfp`** is converted to 1)
- Convert +∞ and -∞ in **`tinyfp`** to **TMin** in integer. (**TMin** represents the smallest integer that can be represented in the 32-bit signed integer format.)
- +NaN and -NaN in **`tinyfp`** are also converted to **TMin** in integer.

2.2.3. `float2tinyfp()`

- A floating-point value that exceeds the range of the **`tinyfp`** representation should be converted to the infinity in **`tinyfp`** (+∞ or -∞ depending on the sign).
- +NaN and -NaN in **`float`** should be converted to the corresponding +NaN and -NaN in **`tinyfp`**, respectively.
- +∞ and -∞ in **`float`** should be converted to the corresponding +∞ and -∞ in **`tinyfp`**, respectively.
- If necessary, use the **round-to-even** mode.

2.2.4. `tinyfp2float()`

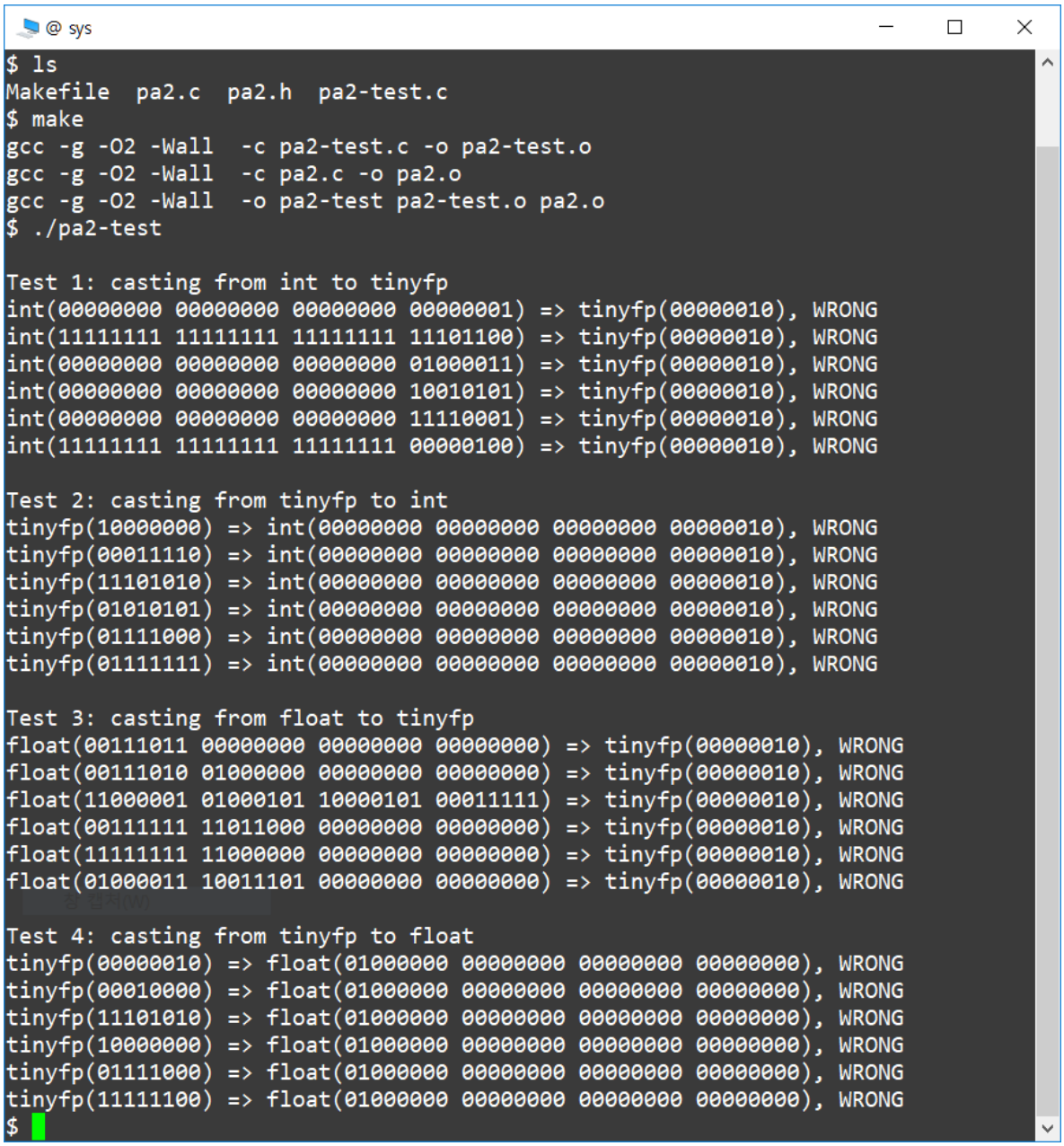
- The **`tinyfp`** type is a subset of the **`float`** type. Hence, all the values in **`tinyfp`** can be represented in the **`float`** format without any error.
- +NaN and -NaN in **`tinyfp`** should be converted to the corresponding +NaN and -NaN in **`float`**, respectively.
- +∞ and -∞ in **`tinyfp`** should be converted to the corresponding +∞ and -∞ in

float, respectively.

- +0 and -0 in **tinyfp** should be converted to the corresponding +0 and -0 in **float**, respectively.

3. Example

The skeleton code will be available in the course homepage at `http://csl.snu.ac.kr`. A simple test code for this project is available in the "`pa2-test.c`" file. Some sample runs look like this:



4. Restrictions

- You should use only the following type variables: **`tinyfp`**, **`float`**, and (singed or unsigned) **`char`** / **`short`** / **`int`**.
- Do not use any array inside **`int2tinyfp()`**, **`tinyfp2int()`**, **`float2tinyfp()`**, and **`tinyfp2float()`** functions.

5. Hand in instructions

- Submit only the **`pa2.c`** file to the submission server.
- **The total number of submissions will be limited to 20 times**. Please debug your code fully before you upload your solution to the server.
- In addition to normal test cases for checking the correctness, we will also measure the time to perform a number of **`float2tinyfp()`** calls. Among the correct implementations, **top 5 fastest solutions will get the 10% bonus.**

6. Logistics

- You will work on this project alone.
- Only the upload submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.
  - You can use up to 5 ***slip days*** during this semester. Please let us know the number of slip days you want to use after each submission.
- Any attempt to copy others' work will result in heavy penalty (for both the copier and the originator). Don't take a risk.

Good luck!

Jin-Soo Kim
Systems Software Laboratory
Dept. of Computer Science and Engineering
Seoul National University