Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2018-10-09
Due: 2018-10-18

ICS 2018 Problem Sheet #5

Problem 5.1: *utopia number system* (3 points)

You have found a calculator from the planet Utopia. You do not know what the number system is that utopia is using, they likely do not use decimal numbers. After playing with the calculator for a while, you have observed the following:

$$\begin{aligned}
\alpha^2 &= \alpha \\
\alpha + \alpha &= \beta \\
\gamma^2 &= \gamma \\
\gamma + \gamma &= \gamma \\
\delta^2 &= \beta\beta \\
\delta + \delta &= \alpha\alpha
\end{aligned}$$

How is the decimal number 99 written on planet Utopia?

Problem 5.2: *b-complement* (1+1 = 2 points)

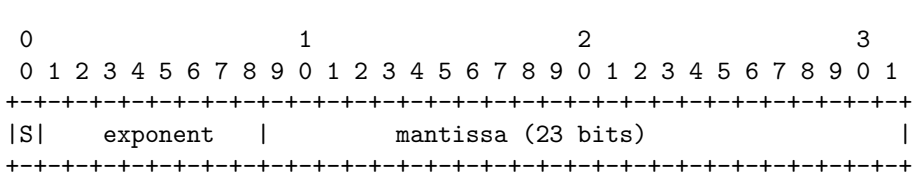
Consider a b-complement number system with the base $b = 5$ and $n = 4$ digits.

a) What is the representation of -1 and -8 in b-complement notation?

b) Add the numbers -1 and -8 in b-complement notation. What is the result in b-complement representation? How is the result converted back into the decimal number system?

Problem 5.3: *IEEE 754 floating point numbers* (3+1 = 4 points)

IEEE 754 floating point numbers (single precision) use the following format (the numbers on the top of the box indicate bit positions, the fields in the box indicate what the various bits mean).



The encoding starts with a sign bit S (set to 1 if the number is negative), followed by the exponent (8 bits), followed by the mantissa (23 bits).

For single-precision floating-point numbers, the exponents in the range of -126 to +127 are biased by adding 127 to get a value in the range 1 to 254 (0 and 255 have special meanings). The exponent itself is represented in binary form (note we have a positive integer after adding the bias).

a) The absolute zero, 0 Kelvin, is at -273.15 degree Celsius. Explain step by step in your own words how the decimal fraction -273.15_{10} is converted into a single precision floating point number.

b) What is the decimal fraction that is actually stored in the single precision floating point number?

Problem 5.4: *decimal to binary (haskell)* (1 point)

Implement a function `bin :: Int -> [Int]` that converts a positive non-zero integer into a list representing the equivalent binary number (a list consisting of 0s and 1s).

```
Prelude> bin 0
[0]
Prelude> bin 5
[1,0,1]
Prelude> bin 42
[1,0,1,0,1,0]
```

Implement a function `binf :: Double -> [Int]` that converts a decimal fraction in the range $[0 \ldots 1)$ into a list representing the equivalent binary fraction (a list consisting of 0s and 1s). (Produce up to 23 binary digits after the dot.)

```
Prelude> binf 0
[0]
Prelude> binf 0.125
[0,0,0,1]
Prelude> binf 0.3359375
[0,0,1,0,1,0,1,1]
Prelude> binf 0.1
[0,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0]
```

The Haskell functions `div`, `mod`, and `truncate` may be helpful.

Submit your Haskell code as a plain text file.