COS10003 Assignment 1

Student Name: Dang Vi Luan

Student ID: 103802759

Question 1:

a/ The conversion of a decimal number $\boldsymbol{x}$ to hexadecimal can be done by repeatedly divide $\boldsymbol{x}$ by $\mathbf{16}$, giving us a quotient $\boldsymbol{q}$ and a remainder $\boldsymbol{r}$, such that $\boldsymbol{x = q * 16 + r}$. We then use $\boldsymbol{r}$ as the least significant digit and generate the remaining digits by repeating the process on $\boldsymbol{q}$ (Bryant & O'Hallaron 2010, p. 74)

$$\mathbf{436 = 27 * 16 + 4\ (4)}$$

$$\mathbf{27 = 1 * 16 + 11\ (\boldsymbol{B})}$$

$$\mathbf{1 = 0 * 16 + 1\ (1)}$$

The hexadecimal representation of $\mathbf{436_{10}}$ is: $\mathbf{1B4_{16}}$.

b/ The handout in week 2 of COS10003 has introduced to us a simple and general understanding of the two's complement representation of negative decimal value.

First, we take the unsigned decimal value and convert it to binary.

$$\mathbf{90_{10} = 01011010_2}$$

Then we flip all the bits.

$$\mathbf{01011010_2 \Rightarrow 1010\ 0101_2}$$

Lastly, we add 1 to the least significant bit.

$$\mathbf{10100101_2 \Rightarrow 1010\ 0110_2}$$

This method works perfectly fine ;however, I would like to explain this result from a mathematical perspective.

For a number $\boldsymbol{x}$ with the length of $\boldsymbol{w}$ and its vector described as $\vec{\boldsymbol{x}} = [\boldsymbol{x}_{\boldsymbol{w-1}}, \boldsymbol{x}_{\boldsymbol{w-2}}, \ldots, \boldsymbol{x}_{\boldsymbol{0}}]$, we have its binary to two's complement representation (hereinafter, referred to as "$\boldsymbol{B2T_w}$") described as (Bryant & O'Hallaron 2010, p. 100):

$$\boldsymbol{B2T_w(\vec{x}) = -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i}$$

Similarly, we can describe $\boldsymbol{-90_{10}}$ as:

$$\boldsymbol{B2T_8([10100110]) = -1.2^7 + 0.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0}$$

$$\Rightarrow \boldsymbol{B2T_8([10100110]) = -128 + 32 + 4 + 2 = -90}$$

Therefore, the two's complement representation of $\boldsymbol{-90_{10}}$ is $\boldsymbol{10100110_2}$.

c/ The standard floating-point format of IEEE consists of 3 components: the Sign, the Exponent ,and the Mantissa. In the context of this question, we will encode 113.75 to IEEE 754 single precision format.

The Sign value of this number will be **0** as the number is positive.

The integer value of this number is $\boldsymbol{113_{10}}$ which is $\boldsymbol{01110001_2}$.in binary which can also be represented as $\boldsymbol{01.10001_2. * 2^6}$**.**

The fraction value of this number is $\boldsymbol{0.75}_{\boldsymbol{10}}$which is $\frac{\boldsymbol{1}}{\boldsymbol{2}} + \frac{\boldsymbol{1}}{\boldsymbol{4}}$, therefore, its binary representation is $\boldsymbol{0.1100_2}$. (Bryant & O'Hallaron 2010, p. 146).

Therefore, $\boldsymbol{113.75_{10}}$ can be represented as $\boldsymbol{01.11000111000000000000000_2 * 2^6}$.

From the integer value and fraction value, we will have **the Mantissa** equal to **11000111000000000000000.**

The Exponent value will be **127 + 6 = 133** which has a binary representation of **10000101**.

Therefore, the IEEE 754 encoding format of $\boldsymbol{113.75_{10}}$ is **0 10000101 11000111000000000000000.**

Question 2:

a/ When we want to convert from a binary value to hexadecimal, apart from transforming it to decimal and from decimal to hexadecimal, we can make use of the relation between binary and hexadecimal formats to make our work easier:

One hexadecimal digit 0 will represent 4 binary zeros (Bryant & O'Hallaron 2010, p. 73).

Binary	**0011**	**0110**	**1101**	**0100**
Hexadecimal	**3**	**6**	**D**	**4**

Therefore, the hexadecimal representation of $\mathbf{0011011011010100_2}$ is $\mathbf{36D4_{16}}$.

b/ The conversion of decimal value $\boldsymbol{x}$ to its binary representation can be done by repeatedly divide $\boldsymbol{x}$ by 2, giving us a quotient $\boldsymbol{q}$ and a remainder $\boldsymbol{r}$, such that $\boldsymbol{x = q * 2 + r}$, We then use $\boldsymbol{r}$ as the least significant binary digit and generate the remaining digits by repeating the process on $\boldsymbol{q}$ (Bryant & O'Hallaron 2010, p. 74)

The conversion of binary value x to its decimal representation can be done reversely by multiplying each of the decimal digits by the appropriate power of $\mathbf{2}$:

$\mathbf{00110110_2} = 2^5 + 2^4 + 2^2 + 2^1 = \mathbf{54_{10}}$

$\mathbf{11010100_2} = 2^7 + 2^6 + 2^4 + 2^2 = \mathbf{212_{10}}$

Therefore, the decimal representation of two 8-bit unsigned integers of $\mathbf{00110110_2}$ is $\mathbf{54_{10}}$,and $\mathbf{11010100_2}$ is $\mathbf{212_{10}}$

c/ As described in part b of question 1, the two 8-bits binary value of $\mathbf{0011011011010100_2}$ can be written as:

$$\boldsymbol{B2T_8}([\mathbf{00110110}]) = -0.2^7 + 2^5 + 2^4 + 2^2 + 2^1 = \mathbf{54_{10}}$$

$$\boldsymbol{B2T_8}([\mathbf{11010100}]) = -1.2^7 + 2^6 + 2^4 + 2^2 = \mathbf{-44_{10}}$$

Therefore, the decimal representation of two 8-bit signed integers (two's complement) of $\mathbf{00110110_2}$ is $\mathbf{54_{10}}$,and $\mathbf{11010100_2}$ is $\mathbf{-44_{10}}$

d/ The IEEE 754 standard specifies a binary16 (IEEE, 2019) as having the following format:

- Sign bit: 1 bit.
- Exponent: 5 bits.
- Mantissa: 10 bits.

In the context of this question, we will translate **0 01101 1011010100** to one half-precision floating point value using the IEEE 754 standard.

As the IEEE floating-point standard represents a number in a form $\mathbf{V} = (\mathbf{-1})^{\boldsymbol{S}} \mathbf{* M * 2^{\boldsymbol{EXP}}}$

Sign is **0**, therefore positive number

Exp = $01101_2 - 15 = 8 + 4 + 1 - 15 =$ **-2**

M = $1.1011010100 = 2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} =$ **1.70703125**

Therefore, the decimal value of **0 011011 011010100** is $(\mathbf{-1})^{\mathbf{0}}$ *** 1.70703125 *** $\mathbf{2^{-2}} = \mathbf{0.4267578125_{10}}$

Question 3:

To decrypt c2 b9 f0 9d 9f ba 39 e2 a0 a1 using UTF-8, let first translate the hexadecimal value to binary as:

11000010 10111001 11110000 10011101 10011111 10111010 00111001 11100010 10100000 10100001

According to the UTF-8 encoding format we can divide the above binary code to 4 characters:

- 11000010 10111001 is 185 which is **UTF+B9**
- 11110000 10011101 10011111 10111010 is 120826 which is **UTF+1D7FA**
- 00111001 is 57 which is **UTF+39**
- 11100010 10100000 1010001 is 5137 which is **UTF+2821**

Once we got the UTF representation of the binary, we can look up the UTF table to find the meaning of the message:

- UTF+B9 is **SUPERSCRIPT ONE**
- UTF+1D7FA is **MATHEMATICAL MONOSPACE DIGIT FOUR**
- UTF+39 is **DIGIT NINE**
- UTF+2821 is **BRAILLE PATTERN DOTS-16**

Therefore, the decrypted message is: ^{1} 𝟺9 ⠡

References

1. Randal E. Bryant and David R. O'Hallaron. 2010, Computer Systems: A Programmer's Perspective, 2nd. ed. . Addison-Wesley Publishing Company, USA.

2. "IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008) , vol., no., pp.1-84, 22 July 2019, doi: 10.1109/IEEESTD.2019.8766229.