**COMP1411 (Spring 2024)  Introduction to Computer Systems**

Individual Assignment 1  Duration: 00:00, 24-Feb-2024 ~ 23:59, 25-Feb-2024

| | |
|---|---|
| *Name* | **SUN Hansong** |
| *Student number* | **23097775D** |

**There are four questions in this assignment (some of the questions have sub-questions). Write down your answers in the blank area under each question. A total of 5 marks are distributed among the questions.**

**For any question, show your steps to obtain the final result. Only giving the final result will cause you to LOSE a significant mark on the questions.**

**Question 1**. [2 marks]

Consider a 32-bit floating-point representation based on the IEEE floating-point format:

- the highest bit is used for the sign bit,
- the sign bit is followed by 5 exponent bits, which are then
- followed by 26 fraction bits.

(1) **Convert** decimal value -29.21875 into the above 32-bit IEEE floating-point format. Write out the result in the hex-decimal format.

*Answer*:

Since -29.21875 < 0, the highest bit is 1.

Bias = 2^(5-1)-1=15

-29.21875 in binary is 11101.00111

So the exponent part is 4+Bias = 19, which is equal to 10011 in binary.

The final result is 11001111101001110000000000000000 in binary

To convert the number in hex-decimal, the result is 0xCFA70000

(2) Assume this 32-bit number is stored on a **big-endian** machine in the addresses 0x100~0x103. Please fill in the following table to show the byte stored in each address. To write a byte, please use the hex-decimal format starting with 0x.

| Memory Address | Byte in the Address |
|---|---|
| 0x0100 | 0xCF |
| 0x0101 | 0xA7 |
| 0x0102 | 0x00 |
| 0x0103 | 0x00 |

**Question 2**. [0.6 marks]

Suppose that x and y are unsigned integers.

**(1) Re-write** the following C-language statement only using << and − operations. Introducing new variables (other than x and y) is not allowed. Please show your steps.

**y = x * 78;**

*Answer*:

Since 78 is equal to 128 – 50

It is 10000000 – 00110010

So, the result is y = (x<<8) – (x<<6) – (x<<5) – (x<<2);

**Question 3**. [1.4 marks]

Consider a 16-bit floating-point representation based on the IEEE floating-point format:

- the highest bit is used for the sign bit,
- the sign bit is followed by 4 exponent bits, which are then
- followed by 11 fraction bits.

(1) What is the **largest positive normalized number** with the above floating-point format? Write the number in binary form.

(2) **Compute** the decimal value of the bit vector 0x6D80 with the above floating-point format. Write the result in decimal format.

*Answer*:

(1)

The largest positive normalized number in the format can be represented by 0111011111111111

And the bias = 2^(4-1) -1 = 7

So, the result in binary is 1.11111111111 * 2 ^ (14-7) = 11111111.1111

(2)

The number 0x6D80 written in binary format is 0110110110000000

Since the highest bit is 0, the number is positive.

The exponential part is 1101, which is 13 in decimal.

So the number in binary is 1.1011 * 2^(13-7) = 1101100

It is 108 in decimal.

**Question 4**. [1 mark]

Given the following C program:

```c
#include "stdio.h"

void main()
{
    unsigned char a;
    char b;

    a = 0x9C;
    b = a;

    printf("b = %d\n", b);
    return;
}
```

(1) What is the output of this program?
(2) Explain why the output is generated in detail.

*Answer*:

(1) The output of this program is b = -100
(2) The data type unsigned char and char contain only 1 byte.
At the beginning of the program, variable a is assigned the value 0x9C, which is equal to 156 in decimal.
Then, b is assigned the value of a, but 156 is bigger than the maximum value of variable b, which is 2^(8-1)-1= 127. So, it causes overflow, which makes the sign bit of b become 1.
This time, the true value of b is equal to –100.
At the end of the program, it converts b into an integer containing 4 bytes, which doesn't cause any error. So, it prints out the true value of b.