COMP1411 (Spring 2022) Introduction to Computer Systems

Individual Assignment 1 Duration: 00:00, 19-Feb-2022 ~ 23:59, 20-Feb-2022

Name	**JIANG GUANLIN**
Student number	**21093962D**

Question 1. [0.5 marks]

Suppose that `x` and `y` are unsigned integers.

Rewrite the following C-language statement by using `<<` and `-`.

```
y = x * 77;
```

Introducing new variables (other than `x` and `y`) is not allowed.

Show your steps. Only giving the final result will NOT get a full mark of this question.

Answer:

y = x * (128 – 32 – 16 – 2 – 1)

y = x * (2^7 - 2^5 – 2^4 – 2^1 – 2^0)

y = (x << 7) - (x << 5) - (x << 4) - (x << 1) - (x << 0);

Question 2. [1 mark]

Suppose that a, b, c and z are all 32-bit unsigned integers.

(1) Assume that the left-most bit is the highest bit. Write C-language statements to set the value of z, such that:
   a. the left-most 10 bits of z are the same as the right-most 10 bits of a;
   b. the right-most 14 bits of z are the same as the left-most 14 bits of b;
   c. the middle 8 bits of z are the same as the right-most 8 bits of c.

Note that:

- You are only allowed to use bit shift operations and logic operations (including bit-wise operators, such as | ^ &) to set the value of z;
- NO arithmetic or if-then-else test (in any form) is allowed;
- Introducing new variables (other than x, y and z) is NOT allowed;
- Using masks is NOT allowed.

(2) If a = 0xC9E3BA75, b = 0x268DBA83, and c = 0x63ABE432, what the be the resulting value of z? Please write the value of z in hex-decimal form starting with prefix 0x.

Show your steps. Only giving the final result will NOT get a full mark of this question.

Answer:

(1)

a = a << (32 - 10) = a << 22

b = b >> (32 - 14) = b >> 18

c = c << (32 - 8) >> (32 - 14 - 8) = c << 24 >> 10

z = a | b | c

(2)

a = 0xC9E3BA75 ➔ 11001001111000111011101001110101 (2)

b = 0x268DBA83 ➔ 00100110100011011011101010000011 (2)

c = 0x63ABE432 ➔ 01100011101010111110010000110010 (2)

After shifting:

a = 10011101010000000000000000000000 (2)

b = 00000000000000000010011010001 1 (2)

c = 00000000000011001000000000000000 (2)

z = 10011101010011001000100110100011 (2) ➔ 0x9D4C89A3 (16)

The answer is z = 0x9D4C89A3

Question 3. [2 marks]

Assume on a big-endian machine, a 32-bit single-precision floating-point number is stored in the addresses 0x0200 ~ 0x0203 is as follows:

Address	Byte in the Address
0x0200	0xC1
0x0201	0x94
0x0202	0x02
0x0203	0x3F

Convert the above floating-point number to a decimal number.

For the converted decimal number, leave only 3 digits after the decimal point and discard all the rest digits; DO NOT write the result in the exponential form of the power of 2 or 10.

Show your steps. Only giving the final result will NOT get a full mark of this question.

Answer:

C194023F ➔ 1 10000011 00101000000001000111111

S = 1 means Negative (-)

Exp = 10000011(2) = 131(10) = Bias + E = 127(10) + E

E = 131 – 127 = 4

M = 1.00101000000001000111111 (2) ➔ 1.15631854534149169922 (10)

v = $(-1)^s$ * M * 2^E

v = $(-1)^1$ * 1.15631854534149169922 * 2^4

v = -18.5010967255 = -18.501

The answer is -18.501

Question 4. [1.5 marks]

Consider a 10-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 5 fraction bits.

Question 1: What is the largest positive normalized number? Write the numbers in both the binary form and the decimal value.

Question 2: **Convert** the decimal number 12.875 into the above 10-bit IEEE floating-point format. Write the result in the binary form.

Show your steps for both Question 1 and Question 2. Only giving the final result will NOT get a full mark of this question.

Answer:

Q1:

Binary: 0 1110 11111

Sign: 0 ➔ + (positive)

Bias = 2 ^ (4 - 1) – 1 = 7

E = Exp – Bias = 1110 (2) – 7 = 14 – 7 = 7

M = 1.11111 (2) = 1.96875 (10)

V = $(-1)^s$ * M * 2^E = 1 * M * 2^7 = 1.96875 * 2^8 = 252

Q2:

12.875 = 2^3 + 2^2 + 2^(-1) + 2^(-2) + 2^(-3)

12.875(10) = 1100.111(2)

The exponent: E = 3

Bias = 2 ^ (k-1) -1 = 2 ^ (4 - 1) – 1 = 7

Exp = Bias + 3 = 7 + 3 = 10(10) = 1010(2)

The sign: S = 0 (positive)

The fraction should be: 1.100111(2)

Without the '1.', so the frac part is 100111. But because of frac part just need 5 bits, this is 6 bits,

We need to be rounding this: 1.100111 → 1.11000

The IEEE single precision float number: 0 1010 11000