CS201
Homework 01

Le Duong Cong Duc
Student ID: 1651044

October 31, 2017

Contents

1 Exercises 1

Write a C expression that will yield a word consisting of the least significant byte of x and the remaining bytes of y.

For operands `x = 0x89ABCDEF` and `y = 0x76543210`, this would give `0x765432EF`.

Answer:

```
/*
 * C expression yeild the least significant byte of x
 * and the remaining bytes of y
 */

(x & 0xFF) | (y & ~(0xFF))
```

2 Exercises 2

Write code for a function with the following prototype:

Answer:

```
/*
 * Mask with least significant n bits set to 1
 * Examples: n = 6 --> 0x3F, n = 17 -->0x1FFFF
 * Assume int is w bits long; 1 <= n <= w;
 */
int lower_one_mask(int n):
{
    return ~((~0) << n);
}
```

3 Exercises 3

Assume that your student ID is a 32-bit integer number. Write the bit sequence of that number in binary. With the same sequence, what is the value (in decimal) if we interpret it as a a) sign-magnitude, b) two's complement?, c) Single precision float

Answer:

3A	Student ID	1651044
3B	Binary	0000 0000 0001 1001 0011 0001 0110 0100
3C	Sign-and-magnitude	1651044
3D	Two's complement	1651044
3E	Single precision float	$412761 * 2^{-147}$

Table 1: Different interpretation

4 Exercises 4

We are running programs on a machine where values of type `int` are 32 bits. They are represented in two's complement, and they are right shifted arithmetically. Values of type `unsigned` are also 32 bits.

We generate arbitrary values x and y, and convert them to unsigned values as follows:

```
/* Create some arbitrary values */
int x = random();
int y = random();
/* Convert to unsigned */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
```

For each of the following C expressions, you are to indicate whether or not the expression always yields 1.

```
A. (x<y) == (-x>-y)
B. ((x+y)<<4) + y-x == 17*y+15*x
C. ~x+~y+1 == ~(x+y)
D. (ux-uy) == -(unsigned)(y-x)
E. ((x >> 2) << 2) <= x
```

Answer:

C expressions	Yes/No
4A	No
4B	Yes
4C	Yes
4D	No
4E	Yes

5 Exercises 5

We are running programs on a machine where values of type int have a 32-bit two's-complement representation. Values of type `float` use the 32-bit IEEE format, and values of type `double` use the 64-bit IEEE format.

We generate arbitrary integer values x, y, and z, and convert them to values of type `double` as follows:

```
/* Create some arbitrary values */
int x = random();
int y = random();
int z = random();
/* Convert to double */
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;
```

For each of the following C expressions, you are to indicate whether or not the expression always yields 1. Note that you cannot use an IA32 machine running gcc to test your answers, since it would use the 80-bit extended-precision representation for both `float` and `double`.

```
A. (float) x == (float) dx
B. dx - dy == (double) (x-y)
C. (dx + dy) + dz == dx + (dy + dz)
D. (dx * dy) * dz == dx * (dy * dz)
E. dx / dx == dz / dz
```

Answer:

C expressions	Yes/No
5A	Yes
5B	No
5C	Yes
5D	No
5E	No

6 Exercises 6

Consider a 16-bit floating-point representation based on the IEEE floating-point format, with 1 sign bit, 5 exponent bits (k = 5), and 10 fraction bits (n = 10). The exponent bias is $2^{5-1} - 1 = 15$.

Fill in the table that follows for each of the numbers given, with the following instructions for each column:

- *Hex*: The four hexadecimal digits describing the encoded form.
- *M*: The value of the significand. This should be a number of the form x or $\frac{x}{y}$, where x is an integer, and y is an integral power of 2. Examples include: 0, $\frac{67}{64}$, and $\frac{1}{256}$.
- *E*: The integer value of the exponent.
- *V* : The numeric value represented. Use the notation x or x x 2^z, where x and z are integers.

Description	Hex	M	E	V
–0	8000	0	-14	-0
Smallest value > 2	4001	$\frac{1025}{1024}$	1	$1025 * 2^{-9}$
512	6000	1	9	512
Largest denormalized	03FF	$\frac{1023}{1024}$	-14	$1023 * 2^{-24}$
Number with hex representation `3BB0`	3BB0	$\frac{123}{64}$	-1	$123 * 2^{-7}$