Homework #5

Due: November 15, 2016 Tuesday

Purpose: This homework is to get you more familiar with bitwise operators.
Carry out the activity of Lab 5 from the textbook.

Assignment:

Read lab 5 and related chapters in textbook Place the code in a5.c.
The code must be modular and properly documented code. Also a Makefile may be used to compile the program (take the one from a4 if exists and modify it).

C-modules to write (place in a5.c):

1. main().
2. int isbigornot(); returns 1 if the underlying architecture uses big-endian representation for multibyte data, 0 otherwise. Follow the suggestion in textbook.
3. int isbigornotalt(); returns 1 if underlying architecture uses big-endian representation for multibyte data, 0 otherwise. It has to different from isbigornot() function. Think.
4. int sizeofint(); returns of size of integer data-type in bytes in the C compiler. Follow the suggestion in textbook.
5. int sizeofintalt(); returns size of integer data-type in bytes in the C-compiler. It has to be different from sizeofint() and should not use sizeof() C-macro.
6. int is2scompornot(); returns 1 if the underlying architecture uses 2's complement representation of integers, 0 otherwise. Hint: think difference in bit-patterns in 2's and 1's complement representations.
7. int is2scompornotalt(); returns if the underlying architecture uses 2's complement representation of integers, 0 otherwise. It has to be different from is2scompornot(). Hint: 1's complement operator in C.

Note: Pay attention to documentation, use make to compile your code (use the one provided with appropriate changes), and check results produced by your program for correct functionality.

Sample output:

```
Big endian architecture:    No
Size of integer in bytes:   9
(Alternate method size):    9
2's complement representation:  No
...
```

Turn-in: a5.c (if no makefile is used.)
A5.tar containing tape archive of a5/Makefile and a5/a5.c
(submit to the BlackBoard)

Lab 5

Representation: Testing Big Endian Vs. Little Endian

Purpose
To learn how the integer representation used by the underlying hardware affects programming and data layout.

Background Reading And Preparation
Read Chapter 3 to learn about big endian and little endian integer representations and the size of an integer.

Overview
Write a C program that examines data stored in memory to determine whether a computer uses big endian or little endian integer representation.

Procedure And Details (checkmark as each is completed)

1. Write a C program that creates an array of bytes in memory, fills the array with zero, and then stores integer 0x04030201 in the middle of the array.

2. Examine the bytes in the array to determine whether the integer is stored in big endian or little endian order.

3. Compile and run the program (without changes to the source code) on both a big endian and little endian computer, and verify that it correctly announces the integer type.

4. Add code to the program to determine the integer size (hint: start with integer 1 and shift left until the value is zero).

5. Compile and run the program (without changes to the source code) on both a thirty-two bit and a sixty-four bit computer, and verify the program correctly announces the integer size.

Optional Extensions (checkmark as each is completed)

6. Find an alternate method of determining the integer size.

7. Implement the alternate method to determine integer size, and verify that the program works correctly.

8. Extend the program to announce the integer format (i.e., one's complement or two's complement).