Name: _____

Experiment 5

C programming to find endian-ness and generate memory dumps

Purpose
Learn to find a machine's endian-ness and to generate memory dumps (print memory contents) used to debug computer systems

Overview
Write a C program that produces a hexadecimal dump of memory in the following format:

Memory Dump (XX-bit EEEEE Endian Machine)
Address Words In Hexadecimal ASCII characters
--------- -------- -------- -------- -------- ----------------
aaaaaaaa xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx cccccccccccccccc

In the above format, each line corresponds to a set of sixteen bytes in memory. The string aaaaaaaa denotes the starting memory address (in hexadecimal) for values on the line, xxxxxxxx denotes the value of a 32-bit item in memory (also in hexadecimal. Note that there are four such 32 bit entities to be printed), and cccccccccccccccc denotes the same sixteen bytes of memory when interpreted as ASCII characters. Note: the ASCII output only displays printable characters; all other characters are displayed as blanks. In the title, XX gives the size of an integer (usually 32 or 64) and EEEEE is either "Big" or "Little", depending on whether the machine is big endian or little endian.

Procedure And Details (check mark as each is completed)
_____ 1. Write a C program that tests and prints the computer's endian-ness and integer size. To test the integer size, place a 1 bit in an integer and count how many times the integer can be shifted left before it becomes zero. To Test endian-ness, place an integer in memory, convert the address to a character pointer, and examine individual bytes.

_____2. To print the hex dump, create a function, mdump that takes two arguments that each specifies an address in memory. The first argument specifies the address where the dump should start, and the second argument specifies the highest address that needs to be included in the dump. Test to ensure that the starting address is less than the ending address. Also make sure that each value specifies an appropriate word address (i.e., an exact multiple of four bytes). For the starting address, round down to the nearest word address; for the ending address, round up.

_____3. Create a structure with the following fields, dump memory in which the structure resides, and find each item in the dump.

1. aa integer 5 (variable name aa, type is integer and value is 5)
2. bb integer 255
3. cc character ’A’
4. dd integer 64206
5. ee char array "Hello World <+>"
6. ff integer 2
7. gg integer 1
8. hh integer 0
9. ii integer -1

If you are unable to show the result in the lab, print the memory dump and circle the contents of the structure. Draw arrows pointing to the contents of the structure and at the other end, write the corresponding field of the structure.

_____ 4. Write a program that builds a singly-linked list, where each node of the list contains an integer. Insert integers 2, 7, 1, 8, and 3 in the list, and write a loop that walks the list to verify that the contents are correct. Find the lowest and highest addresses used in the list, and display a hex dump of the memory locations. Save the output of mdump (as an image like .bmp) for the structure. Using any image editor (like MS Paint), draw arrows to identify the nodes of the linked list.