**Lab-1 Little Endian, C Program and Tools**

Full Score: 10 pts | Due: Wednesday 9/23

**Verify little endian byte order of ARM machine. Write a C program [name it as LastnameFirstInitial\_l1.c (e.g., if your name is John Smith, it will be SmithJ\_l1.c)] using only main function to accomplish the following tasks:**

1. Declare an unsigned int variable x and an unsigned int pointer variable px which holds the address of variable x.
2. Declare an int pointer variable py, which points to an integer size of dynamic (heap) memory allocated by malloc.
3. Prompt user entering the unsigned int 0xaabbccdd in hexadecimal form into x by using printf and scanf.
4. Prompt user entering the int 0x12345678 in hexadecimal form into \*py by using printf and scanf.
5. Declare a char pointer pc and an int variable i for loop index use.
6. To double check user inputs print out values in x, px, \*py, and py with printf function using appropriate format strings.
7. Let pc point to variable x, then use a for-loop to print out the value of pc and \*pc (the char value the pc points to). Increase pc by 1, i.e., pc++, before next iteration. This will show that 0xdd is stored in lowest address and 0xaa in highest address, exactly the way little-endian byte order for unsigned int type variable x.
8. Let pc point to variable px, i.e., pc = &px, then use a for-loop to print out values of (pc + i) and \*(pc + i), i.e., the byte values of content of px (the address of x). This time pc is fixed, but by adding loop variable i into pc makes (pc + i) points to next char. This will show that the 8 bytes address value of also is stored in little-endian order.
9. Repeat steps 7 and 8 with pc points to py and address of py, i.e. pc = &py. You should notice that dynamic (or heap) memory addresses are much smaller than that of local variables (they are stored in stack memory).
10. De-allocate the dynamic memory pointed by py with free.

After having your SmithJ\_l1.c written in any text editor (on Raspberry Pi, you can use Geany or vi), go to Terminal to compile it with gcc (GNU C Compiler) as: gcc –g SmithJ\_l1.c. If there is no syntax error, you should be able to see an executable file called a.out generated in the same directory (or folder) as your c file. You can run your program in the Terminal by typing ./a.out followed by enter to get user inputs and outputs. Then use the debugger GDB to run the program as gdb ./a.out. In gdb, set a breakpoint at main function, then enter run to execute the program line by line. You should use print (or its abbreviation p) command to print out values of your variables in multiple formats.

**Submission**: Upload your source file LastnameFirstInitial\_l.c (e.g., if your name is John Smith, it will be SmithJ\_1.c) and running results (can be appended at the end of .c file) to your blackboard account by the due day.