**Material:**  Lectures 2-3
**Due**:  Start of Lecture 5:  Tuesday, 8 June by 2pm EDT

**Submission notes:**
- For full credit, please show your work and denote your answers with a circle or a box.
- Always write and draw your diagrams neatly!  We cannot be expected to GUESS what you meant to write!
- Please see the submission guidelines on the homework page of the course website for details.

1. (5 pts) If you have not done so already, please complete the following surveys to tell us a bit of information about you and your status with the lab materials, which will help us plan the course:
   a. Course background survey:  https://wpi.qualtrics.com/jfe/form/SV_cBDMgYGeMVFHvD0
   b. Lab parts survey:  https://wpi.qualtrics.com/jfe/form/SV_6DswHL0bfuJ0nhY

   You do not need to submit anything for this part—just fill out the surveys.  If you have already completed the surveys, then you are already done with this part!  If you have trouble completing the surveys, please let us know.

2. (5 pts) Answer the following questions regarding execution of loops.  Assume that the code below is executed on an MSP430 (ie, an `int` has a size of two bytes).

   a. In the following code segment, how many times does the loop iterate?  What are the values of `kk`, `mm`, and `inVal` at the end of the first and last loops?  (You can specify the values in either hex or decimal, whichever you prefer.)

```
unsigned int inVal, out, kk = 0x0001, mm = 0x1000;
char count = 1;

// ...

while(kk != 0)
{
    inVal = (mm + kk) / kk;
    // . . .

    mm /= 2;
    kk = kk << 1;
}

// ...
```

**(continued on the next page)**

**3.** (10 pts) In a table like the one below, show how the following variables would be stored successively in memory by:
- A Little-endian CPU
- A Big-endian CPU

```
unsigned long ser_num = 0xFEEDCAFE; // An unsigned 32-bit integer
int ii = 65;                // Two's complement 16-bit integer
char class[8] = "ECE2049"; // Null-terminated string containing ASCII text
int arr[2] = {255, -1};    // 2-element array of int
unsigned long count = 16322;
```

  **a.** First, determine the representation of each variable listed above in hex.

  **b.** Next, arrange the variables as they would be stored in memory starting at address 0x4200. To do this, make a table similar to the one below, labeling each address and denoting which byte would be stored in that address in each case. Remember that each address in memory holds 1 byte.

| Address | Little Endian | Big Endian |
|---------|---------------|------------|
| … | | |
| … | | |
| … | | |
| 0x4202 | | |
| 0x4201 | | |
| 0x4200 | | |

  Note: This table is usually drawn with the lowest memory address at the *bottom* of the table, as shown here:

  **c.** Is the MSP430 architecture we use in class big or little endian?

  **d.** What if we added one additional variable in the example above as shown below. Based on only the code below, what is the value of `new_var`? How is it arranged in memory?

```
// ...
long unsigned int count = 16322; // Last variable from above
unsigned int new_var;
```

**4.** (5 pts) **Debugging challenge:**  Consider the code example below.  This program is designed to compare the contents of two arrays:  if they are equal, the program should display "TRUE," if the arrays are not equal, the program should display "FALSE".
However, even when the arrays have the same value, the program always prints "FALSE", regardless of the values in arrays a and b.

What is going on here?  What about the program causes the comparison to always fail?

```
int a[4] = {1, 2, 3, 4};
int b[4] = {1, 2, 3, 4};

if (a == b) {
    display("TRUE");
else {
    display("FALSE");
}
```

If you would like to run this program (not required), you can find a CCS project containing this example listed with the homework document on the course website.  To run the project, download and import the zip file into CCS in the same manner as you did for the lab 0 demo project (see the lab 0 assignment for instructions).