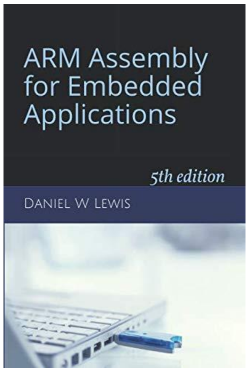


Programming Lab 2A

8-Bit Binary Numbers

Click to download Lab2A-Main.c

Topics: Decimal to binary conversion; unsigned and 2's complement signed interpretation of binary numbers; overflow; binary addition

Prerequisite Reading: Chapters 1-2 & Appendix B
Revised: December 26, 2020

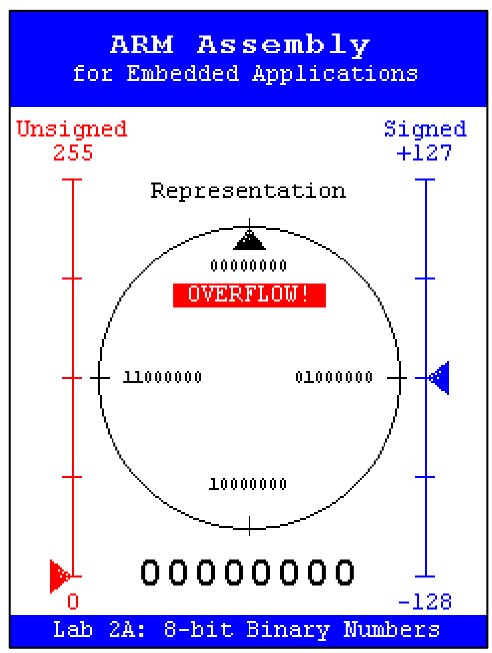
Background: This lab exercises your understanding of binary number systems. Since you have not been introduced to assembly language yet, this assignment is to be coded entirely in C. Successful completion of this assignment will also reinforce your familiarity with the workspace environment.

Assignment:

1. Delete any existing files in the `src` and `obj` subdirectories of your workspace folder.
2. Click on the link above to download the C main program and store it in the `src` subdirectory of your workspace folder.
3. Use your favorite text editor (<u>not</u> a word processor) to create a second C source code file in the `src` subdirectory that implements the three functions shown below. Do not use filenames containing spaces or filename extensions with uppercase letters. Each array parameter holds an 8-bit binary number, $b_7b_6b_5b_4b_3b_2b_1b_0$, where `bits[7]` = b_7 and `bits[0]` = b_0.

```
int32_t  Bits2Signed(int8_t bits[8]) ;     // Convert array of bits to signed int.
uint32_t Bits2Unsigned(int8_t bits[8]) ; // Convert array of bits to unsigned int
void     Increment(int8_t bits[8]) ;       // Add 1 to value represented by bit pattern
void     Unsigned2Bits(uint32_t n, int8_t bits[8]) ; // Opposite of Bits2Unsigned.
```

When the program runs, it should cycle through all the 8-bit patterns in sequence, displaying the bit pattern of the representation, as well as its interpretation as both unsigned and signed 2's complement integers. If there is an error in one of your functions, the program will display your output in white text on a red background and halt.



Hint: The following is the most efficient way to convert binary to decimal: Consider an 8-bit binary signed integer, represented as $b_7b_6b_5b_4b_3b_2b_1b_0$, where the b's are the 1's and 0's of the number. The corresponding polynomial would be:

$$n = 2^7b_7 + 2^6b_6 + 2^5b_5 + 2^4b_4 + 2^3b_3 + 2^2b_2 + 2^1b_1 + 2^0b_0$$

But note that you can rewrite this as:

$$n = b_0 + 2(b_1 + 2(b_2 + 2(b_3 + 2(b_4 + 2(b_5 + 2(b_6 + 2b_7)))))))$$

Which can be computed using a simple loop:

$$n \leftarrow 0$$
$$for\ i = 7\ down\ to\ 0:$$
$$\quad n \leftarrow 2 \times n + b_i$$