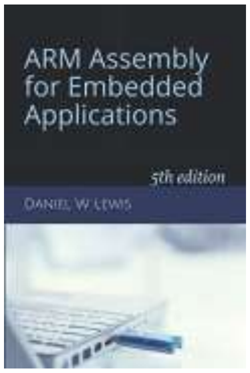


Programming Lab 11F

Converting Reals

Click to download Lab11F-Main.c

Topics: Q16 fixed-point and single-precision float representation formats, bit manipulation.

Prerequisite Reading: Chapters 1-11

Revised: January 1, 2021

Background: Q16 and IEEE single-precision floating point are two different 32-bit representations of real numbers. The objective of this lab is to create functions in assembly that can be used to convert the representation of a variable from one of these formats to the other. For a processor without floating-point instructions, such functions must be implemented using only integer and bit-manipulation instructions.

The range of a Q16 representation is limited to $\pm 2^{15}$(or about $\pm 3.3 \times 10^5$), while that of single-precision floating-point is much greater at $\pm 2^{127}$ (or about $\pm 3.4 \times 10^{38}$). However, Q16 can represent (large) numbers with as many as 10 significant decimal digits (five on either side of the decimal point), while all floating-point numbers are limited to no more than eight significant digits. As a result, converting some Q16 numbers to floating-point will result in the loss of some least-significant digits. Converting a floating-point number to Q16 never causes a loss of significant digits, but may exceed the smaller range the Q16 representation.

Assignment: The main program includes the following two functions written in C to perform such conversions.

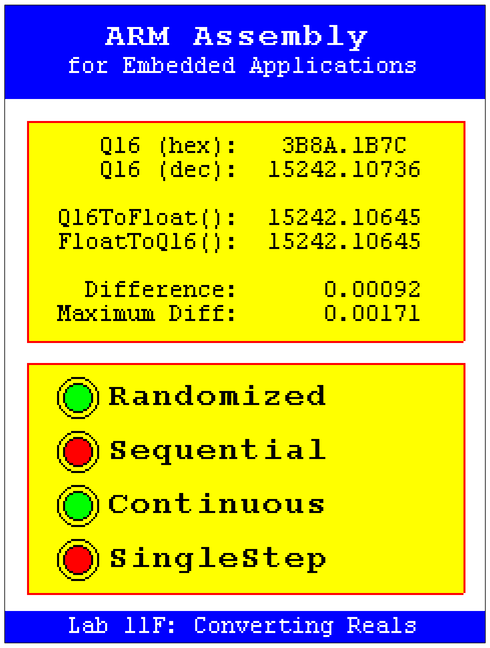
```
typedef int32_t Q16 ;         // These data types are defined as 32-bit integers so they
typedef int32_t FLOAT ;       // can be manipulated using bit-manipulation instructions.

FLOAT Q16ToFloat(Q16 q) ;     // Both the parameter and return value of each of
Q16 FloatToQ16(FLOAT f) ;     // these functions are held in register R0.
```

You can compile and run the program as is without writing any assembly. However, your task is to create faster version of these functions in assembly using the C versions to guide your implementation. Do not use any floating-point instructions! The original C versions have been defined as "weak", so that the linker will automatically replace them in the executable image by those you create in assembly; there is no need to remove the C versions.

These functions call three other functions that are not needed in assembly: every call to C functions `CLZ`, `UBFX`, or `BFI` may be replaced in your assembly by a single `CLZ, UBFX or BFI` instruction. These three C functions are also defined as weak, which allows the linker to eliminate them from the executable image if not used.

Suggestion: If your code is correct, the display should look like the figure shown on the right. The program only tests using numbers within the Q16 range of values. Any incorrect conversions are displayed as white text on a red background. (Note: Touching the radio buttons changes the test mode as indicated; when in single step mode, pressing the blue pushbutton advances to the next test.)



```
ARM Assembly
for Embedded Applications

   Q16 (hex):   3B8A.1B7C
   Q16 (dec):  15242.10736

Q16ToFloat():  15242.10645
FloatToQ16():  15242.10645

  Difference:      0.00092
Maximum Diff:      0.00171
```