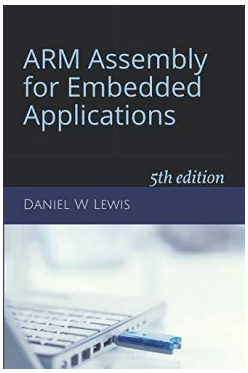


Programming Lab 10A

Solving Quadratics with Software Floating-Point

Click to download Lab10A-Main.c

Click to download real-libs.zip

Topics: Floating-point emulation

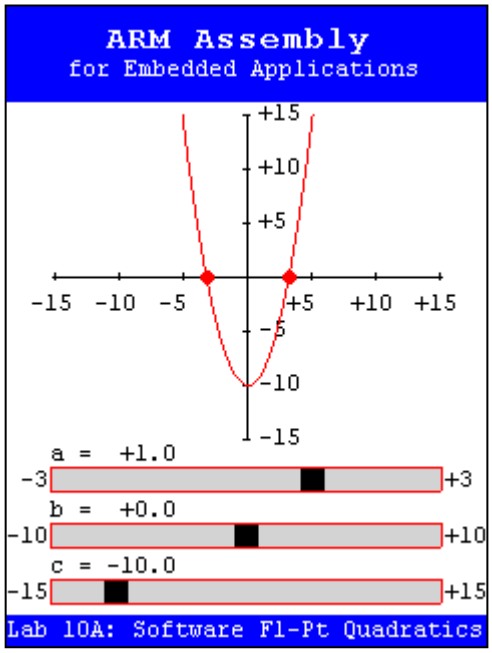
Prerequisite Reading: Chapters 1-10

Revised: May 27, 2021

Background: This assignment is identical to Lab 9A except that instead of hardware floating-point instructions, your code will use a software floating-point emulation library. The library uses 32-bit integers to hold the bit patterns of floating-point values:

```
typedef int32_t float32_t ;
```

Assignment: The main program will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following four functions found in the C main program. 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; you do not need to remove the C versions. This allows you to create and test your assembly language functions one at a time.



`float32_t Root1(float32_t a, float32_t b, float32_t c) ;`

Computes the root given by $\frac{-b+\sqrt{Discriminant(a,b,c)}}{2a}$

`float32_t Root2(float32_t a, float32_t b, float32_t c) ;`

Computes the root given by $\frac{-b-\sqrt{Discriminant(a,b,c)}}{2a}$

`float32_t Quadratic(float32_t x, float32_t a, float32_t b, float32_t c) ;`

Computes the quadratic, $ax^2 + bx + c$ *Note: The most efficient implementation is* $c + x(b + ax)$

`float32_t Discriminant(float32_t a, float32_t b, float32_t c) ;`

Computes the value of the discriminant, $b^2 - 4ac$ *Note: Functions Root1 and Root2 should call this function.*

Note that your assembly language code must call several C functions (shown below) from a floating-point emulation library. Your code will likely need to push and pop several registers; be sure that the total number of registers you push and pop is even so that the address held in the stack pointer remains a multiple of eight to satisfy the data alignment requirements of those library functions.

Download the main program and ZIP file containing the floating-point emulation libraries. Extract file `lib2-float.c` to the `src` directory of your workspace. Use the following functions to perform arithmetic:

```
float32_t AddFloats(float32_t a, float32_t b) ;        // returns a + b
float32_t SubFloats(float32_t a, float32_t b) ;        // returns a - b
float32_t MulFloats(float32_t a, float32_t b) ;        // returns a * b
float32_t DivFloats(float32_t a, float32_t b) ;        // returns a / b
float32_t SqrtFloat(float32_t x) ;                     // returns sqrt(x)
```

Test your program with the main program. If your code is correct, the display should look similar to the image shown, the sliders can be used to vary the coefficient values, and pressing the blue pushbutton will restore the initial conditions. Otherwise, incorrect return values will cause an error message to be displayed as white text on a red background and the program will be halted.