


## Programming Lab 10B


# Solving Quadratics with Software Posit Reals

Topics: Representation of real numbers using posit emulation.

Prerequisite Reading: Chapters 1-10

Revised: June 22, 2021

 Click to download Lab10B-Main.c

 Click to download real-libs.zip

**Background<sup>1</sup>:** This assignment is identical to Lab 9A except that instead of hardware positing-point instructions, your code will use a software posit emulation library. The library uses 32-bit integers to hold the bit patterns of posit values:

```
typedef int32_t posit32_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.

```
posit32_t Root1(posit32_t a, posit32_t b, posit32_t c) ;
```

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

```
posit32_t Root2(posit32_t a, posit32_t b, posit32_t c) ;
```

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

```
posit32_t Quadratic(posit32_t x, posit32_t a, posit32_t b, posit32_t c) ;
```

Computes the quadratic,  $ax^2 + bx + c$   
Most efficiently implemented as  $c + x(b + ax)$

```
posit32_t Discriminant(posit32_t a, posit32_t b, posit32_t c) ;
```

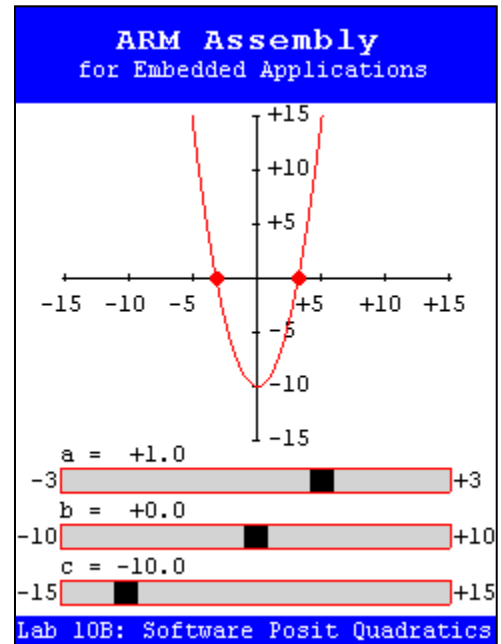
Computes the value of the discriminant,  $b^2 - 4ac$   
Functions Root1 and Root2 should call this function.

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 the ZIP file containing the emulation libraries. Extract file lib3-posit.c to the src directory of your workspace. Use the following functions to perform arithmetic:

```
posit32_t AddPosits(posit32_t a, posit32_t b) ; // returns a + b
posit32_t SubPosits(posit32_t a, posit32_t b) ; // returns a - b
posit32_t MulPosits(posit32_t a, posit32_t b) ; // returns a * b
posit32_t DivPosits(posit32_t a, posit32_t b) ; // returns a / b
posit32_t SqrtPosit(posit32_t x) ; // returns sqrt(x)
posit32_t Float2Posit(float f32) ; // converts float to posit
```

Test your code 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.



<sup>1</sup> [https://en.wikipedia.org/wiki/Unum\\_\(number\\_format\)#Unum\\_III](https://en.wikipedia.org/wiki/Unum_(number_format)#Unum_III)