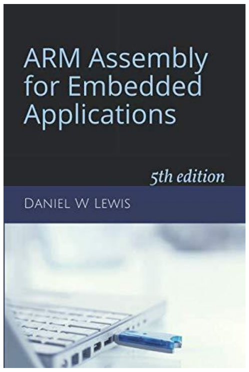


Programming Lab 11C

Implementing Division for Q16 Fixed-Point Reals

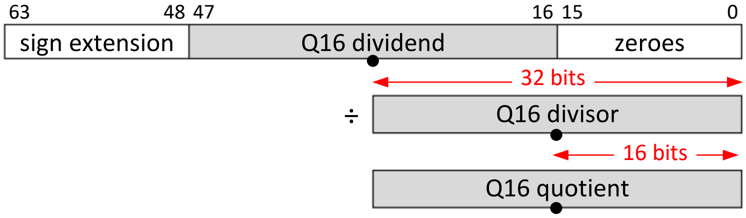
Click to download Lab11C-Main.c

Topics: Representation of real numbers using Q16 fixed-point.

Prerequisite Reading: Chapters 1-11

Revised: June 7, 2021

Background: Q16 division requires that the 32-bit Q16 dividend be positioned in the middle of a 64-bit integer and sign-extended so that the imaginary binary point will be in the middle of the resulting quotient. Unfortunately, the ARM integer divide instructions only support a 32-bit dividend.



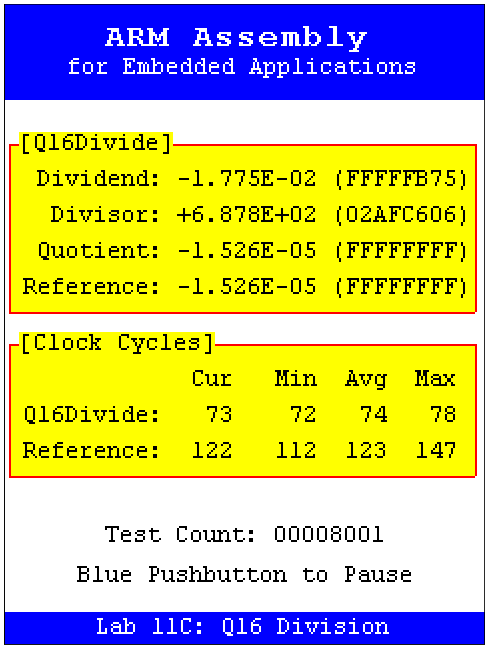
Writing a function for 64÷32 division normally requires a loop of 32 iterations – once for every bit in the divisor. However, a loop of 16 iterations is sufficient to convert the quotient of a `UDIV` instruction to an *unsigned* Q16 quotient as shown in the adjacent pseudocode. This may be used as the basis of a routine to produce a *signed* Q16 quotient from the absolute values of its operands by handling the sign of the quotient separately.

```
repeat 16 times:
  {
  quotient ← 2 × quotient
  remainder ← 2 × remainder
  if (remainder ≥ divisor)
     {
     remainder ← remainder − divisor
     quotient ← quotient + 1
     }
  }
```

Assignment: The main program includes a C function `Q16Divide` that uses this approach to implement Q16 division. You can compile and run the program as is without writing any assembly. However, your task is to create a faster version of `Q16Divide` in assembly using the C version to guide your implementation. The original C version has been defined as "weak", so that the linker will automatically replace it in the executable image by the one you create in assembly; there is no need to remove the C version.

Since the objective of implementing the function in assembly is speed, you are to avoid branch instructions. Use the `.rept` and `.endr` directives to "unroll" the loop, use bitwise operations to change the sign of a value, and implement simple decisions using `IT` instructions instead of a `CMP` and conditional branch wherever possible.

The main program repeatedly calls your `Q16Divide` function with randomly selected dividends and divisors and compares the quotient and execution time to that of a reference version written entirely in C[1]. Updates to the display will pause on any error or while the blue push-button is pressed. Errors are displayed as white text on a red background.



```
         ARM Assembly
   for Embedded Applications

[Q16Divide]
  Dividend: -1.775E-02 (FFFFFB75)
   Divisor: +6.878E+02 (02AFC606)
  Quotient: -1.526E-05 (FFFFFFFF)
 Reference: -1.526E-05 (FFFFFFFF)

[Clock Cycles]
             Cur    Min   Avg   Max
 Q16Divide:   73     72    74    78
 Reference:  122    112   123   147

       Test Count: 00008001
    Blue Pushbutton to Pause

      Lab 11C: Q16 Division
```

[1] The C version sign-extends the dividend to 64 bits, shifts it left by 16 bits, and then divides by the 32-bit divisor. C promotes the divisor to a 64-bit integer to match the data type of the dividend, which results in a library function call to perform 64÷64 division.