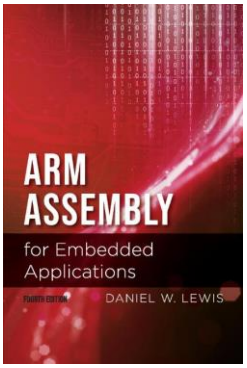

Programming Lab #7b

Multiplication by a Constant

Prerequisite Reading: Chapters 1-8
Revised: February 22, 2019

Background: Our Cortex-M4F processor has a very fast multiply instruction, but many inexpensive processor chips either do not, or have one that is very slow. Therefore, in this assignment you will gain experience finding minimal length sequences of addition, subtraction and left shifts to multiply by a constant. A simple approach is to decompose the multiplier into powers of 2. For example, to multiply variable x in register `R0` by 60_{10}, we decompose 60x into $4x + 8x + 16x + 32x$, which leads to the following instruction sequence:

```
MOV R1,R0,LSL 2          // R1 = 4*R0
ADD R1,R1,R0,LSL 3       // R1 = 4*R0 + 8*R0
ADD R1,R1,R0,LSL 4       // R1 = 4*R0 + 8*R0 + 16*R0
ADD R0,R1,R0,LSL 5       // R0 = 4*R0 + 8*R0 + 16*R0 + 32*R0
```

However, the binary representation of 60 (i.e., 00111100_2) contains a string of four 1's, which may be expressed as the difference of two powers of 2, leading to a two-instruction solution. Although there is no simple procedure that can determine the shortest sequence of instructions, sometimes factoring helps – especially for large constants. For example, if M can be factored as $M_1 x M_2$, then it may be better to implement a sequence for M_1 and then use that result in a sequence that multiplies by M_2.

```
 01000000
-00000100
 00111100
```

Assignment: Create the following five ARM assembly language functions:

```
uint32_t  Times45(uint32_t mcnd) ;      // returns 45 * mcnd
uint32_t  Times55(uint32_t mcnd) ;      // returns 55 * mcnd
uint32_t  Times106(uint32_t mcnd) ;     // returns 106 * mcnd
uint32_t  Times43691(uint32_t mcnd) ;   // returns 43691 * mcnd
```

Your challenge is to create solutions that multiply correctly without loops, conditional branches, `IT` blocks, multiply instructions or .rept macros, and using the least number of clock cycles and registers that you can.

Test your code using the C main program found here. The main program will choose random test values for the function parameters. If your code computes the product correctly, the display should look similar to the image to the right. Incorrect products will be displayed as white text on a red background. And yes, the clock cycles shown are in fact possible. 😊

Note: *The clock cycle counts that are displayed do not include the time required to load parameters into registers, or for the function call (BL) or function return (BX). The clock cycle counts are only those used "inside" the function. Since the ARM add, subtract and shift instructions all execute in one clock cycle, the clock cycle counts shown at right happen to also be the same as the number of instructions in the optimal solution.*