Multiplying by a small constant is often faster using shifts and additions.

To use shifts and addition (or subtraction):

1) convert the constant into a sum of powers of two.

2) convert the multiplications by powers of two into left shifts.
Example 1: $x \times 10$

1) convert the constant into a sum of powers of two

$$x \times 10 = x \times (8 + 2) = (x \times 2^3) + (x \times 2^1)$$

2) convert the multiplications by powers of two into left shifts

$$x \times 10 = x \times (8 + 2) = (x \times 2^3) + (x \times 2^1) = (x << 3) + (x << 1)$$
Example 1 cont'd:

\[ x \times 10 = x \times (8 + 2) = (x \times 2^3) + (x \times 2^1) \]
\[ = (x \ll 3) + (x \ll 1) \]

ARM code: Put x in r0  
\text{(NOTE: do not destroy r0)}

\begin{align*}
\text{mov r1, r0, lsl #3} & \quad // r1 = r0 \times 8 \\
\text{add r1, r1, r0, lsl #1} & \quad // r1 = r1 + r0 \times 2
\end{align*}

Why have an \textit{add} instruction in the above example? Because only the second source operand can be shifted.
Example 2: \( x \times 13 \)

1) convert the constant into a sum of powers of two
\[
x \times 13 = x \times (8 + 4 + 1)
\]
\[
= (x \times 8) + (x \times 4) + (x \times 1)
\]
\[
= (x \times 2^3) + (x \times 2^2) + (x \times 2^0)
\]

2) convert the multiplications by powers of two into left shifts
\[
x \times 13 = (x \times 2^3) + (x \times 2^2) + x
\]
\[
= (x << 3) + (x << 2) + x
\]
Multiplication by small constants (pp. 139 – 140)

Example 2 cont'd:
\[ x \times 13 = (x \times 2^3) + (x \times 2^2) + (x \times 2^0) \]
\[ = (x << 3) + (x << 2) + x \]

ARM code: Put x in r0 \hspace{1cm} (NOTE: do not destroy r0)
add r1, r1, r0, lsl #3
add r1, r1, r0, lsl #2
add r1, r1, r0
Example 3: \( x \times 30 \)

1) convert the constant into a sum of powers of two

\[
x \times 30 = x \times (16 + 8 + 4 + 2) = (x \times 16) + (x \times 8) + (x \times 4) + (x \times 2) = (x \times 2^4) + (x \times 2^3) + (x \times 2^2) + (x \times 2^1)
\]

2) convert the multiplications by powers of two into left shifts

\[
x \times 30 = (x \times 2^4) + (x \times 2^3) + (x \times 2^2) + (x \times 2^1) = (x \ll 4) + (x \ll 3) + (x \ll 2) + (x \ll 1)
\]
Multiplication by small constants (pp. 139 – 140)

Example 3 cont'd:

\[ x \times 30 = (x \times 2^4) + (x \times 2^3) + (x \times 2^2) + (x \times 2^1) \]

\[ = (x << 4) + (x << 3) + (x << 2) + (x << 1) \]

ARM code: Put x in r0  

\textbf{(NOTE: do not destroy r0)}

\begin{verbatim}
  mov r1, r0, lsl #4
  add r1, r1, r0, lsl #3
  add r1, r1, r0, lsl #2
  add r1, r1, r0, lsl #1
\end{verbatim}
Example 4: $x * 7$

$x * 7 = x * (8 - 1) = (x * 8) - (x * 1)$

$= (x * 2^3) - (x * 2^0)$

$x * 15 = (x * 2^3) - (x * 2^0)$

$= (x << 3) - x$
Example 4 cont'd:
\[ x \times 7 = (x \times 2^3) - (x \times 2^0) \]
\[ = (x \ll 3) - x \]

ARM code: Put x in r0
```
rsb r1, r0, r0, lsl #3       // r1 = 8*r0 - r0 = 7*r0
```

rsb rd, rn, op2 (reverse subtract): subtracts rn from op2;

therefore,
```
rsb r1, r0, r1       is the same as
sub r1, r1, r0       // r1 = r1 - r0
```
Division by small constants (p. 108)

\[
\text{MOV } r1, r3, \text{ ASR } #7 \quad // \quad r1 = r3/128
\]

vs.

\[
\text{MOV } r1, r3, \text{ LSR } #7 \quad // \quad r1 = r3/128
\]

The first treats the registers like signed values (shifts in MSB). The latter treats data like unsigned values (shifts in 0).

\text{int vs unsigned int} \gg