Listings 2.3 and 2.4 show the register R0 containing a decimal 17 being shifted 2 bits to the left using an immediate shift count or a shift count in a register, respectively. Both cases will result in R0 containing a value of decimal 68. Listing 2.3 demonstrates the Logical Shift Left (LSL) instruction, but I suggest you also try LSR and ROR (rotate) instructions.

```
            .global     _start      @ Provide program starting address to linker
_start:     mov         R0,#17      @ Use 17 for test example (could be anything)
            lsl         R0,#2       @ Shift R0 left 2 bits (i.e., multiply by 4)
            mov         R7,#1       @ Service command code 1 terminates this program.
            svc         0           @ Issue Linux command to terminate program
            .end
```

Listing 2.3: Shifting a register's contents is like multiplying by a power of 2.

```
            .global     _start      @ Provide program starting address to linker
_start:     mov         R0,#17      @ Use 17 for test example (could be anything)
            mov         R6,#2       @ A second integer for test
            lsl         R0,R6       @ Shift R0 left by the value in R6 (i.e., multiply by 4)
            mov         R7,#1       @ Service command code 1 terminates this program.
            svc         0           @ Issue Linux command to terminate program
            .end
```

Listing 2.4: Shifting a register by the value in another register

One final note about shift instructions on the ARM processor. Their assembly language coding might look similar to that of other CPUs, but the code generated has a surprise to be revealed in Lab 5.