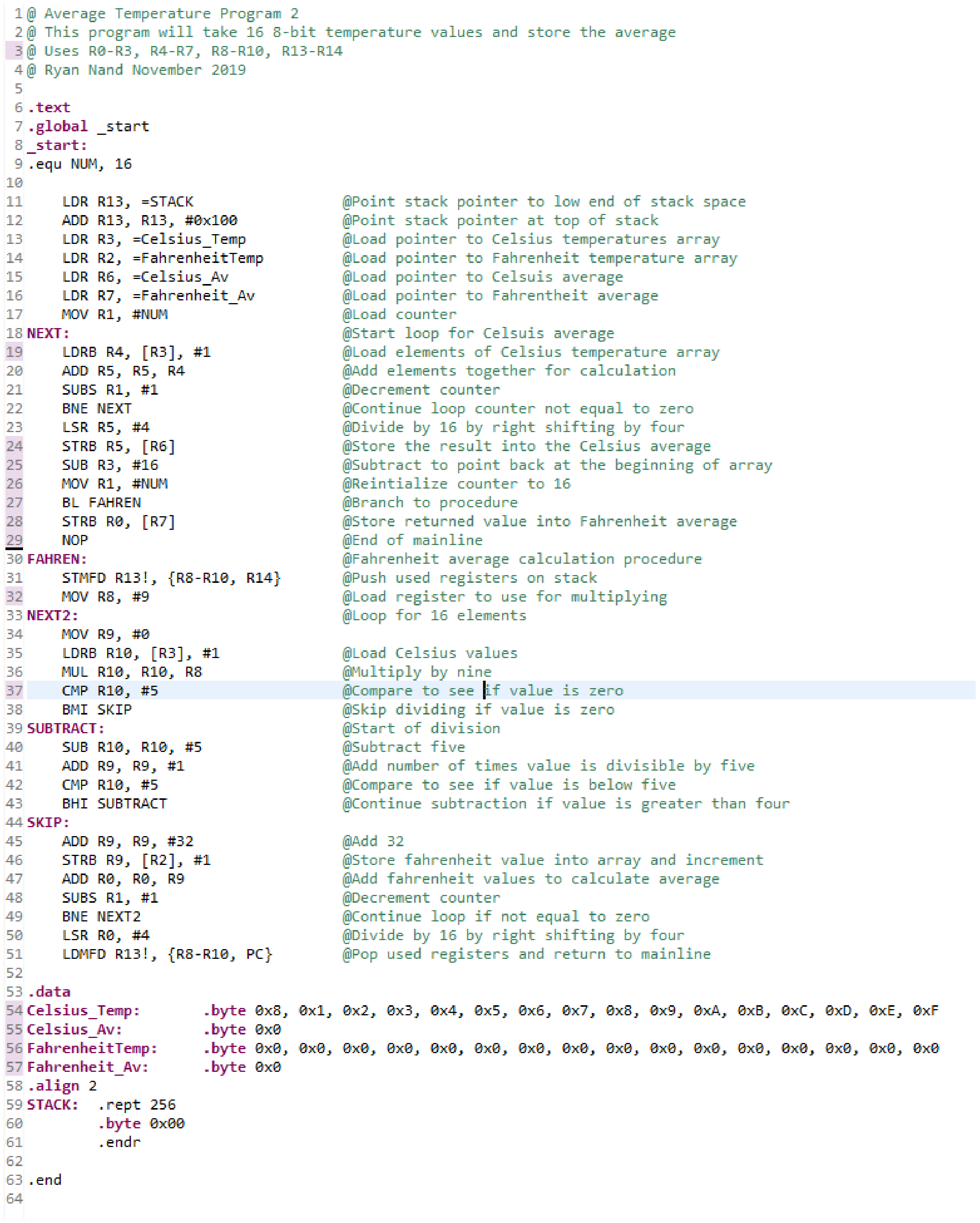

```
@ Average Temperature Program 2
@ This program will take 16 8-bit temperature values and store the average
@ Uses R0-R3, R4-R7, R8-R10, R13-R14
@ Ryan Nand November 2019

.text
.global _start
_start:
.equ NUM, 16

    LDR R13, =STACK                 @Point stack pointer to low end of stack space
    ADD R13, R13, #0x100            @Point stack pointer at top of stack
    LDR R3, =Celsius_Temp           @Load pointer to Celsius temperatures array
    LDR R2, =FahrenheitTemp         @Load pointer to Fahrenheit temperature array
    LDR R6, =Celsius_Av             @Load pointer to Celsuis average
    LDR R7, =Fahrenheit_Av          @Load pointer to Fahrentheit average
    MOV R1, #NUM                    @Load counter
NEXT:                               @Start loop for Celsuis average
    LDRB R4, [R3], #1               @Load elements of Celsius temperature array
    ADD R5, R5, R4                  @Add elements together for calculation
    SUBS R1, #1                     @Decrement counter
    BNE NEXT                        @Continue loop counter not equal to zero
    LSR R5, #4                      @Divide by 16 by right shifting by four
    STRB R5, [R6]                   @Store the result into the Celsius average
    SUB R3, #16                     @Subtract to point back at the beginning of array
    MOV R1, #NUM                    @Reintialize counter to 16
    BL FAHREN                       @Branch to procedure
    STRB R0, [R7]                   @Store returned value into Fahrenheit average
    NOP                             @End of mainline
FAHREN:                             @Fahrenheit average calculation procedure
    STMFD R13!, {R8-R10, R14}       @Push used registers on stack
    MOV R8, #9                      @Load register to use for multiplying
NEXT2:                              @Loop for 16 elements
    MOV R9, #0
    LDRB R10, [R3], #1              @Load Celsius values
    MUL R10, R10, R8                @Multiply by nine
    CMP R10, #5                     @Compare to see if value is zero
    BMI SKIP                        @Skip dividing if value is zero
SUBTRACT:                           @Start of division
    SUB R10, R10, #5                @Subtract five
    ADD R9, R9, #1                  @Add number of times value is divisible by five
    CMP R10, #5                     @Compare to see if value is below five
    BHI SUBTRACT                    @Continue subtraction if value is greater than four
SKIP:
    ADD R9, R9, #32                 @Add 32
    STRB R9, [R2], #1               @Store fahrenheit value into array and increment
    ADD R0, R0, R9                  @Add fahrenheit values to calculate average
    SUBS R1, #1                     @Decrement counter
    BNE NEXT2                       @Continue loop if not equal to zero
    LSR R0, #4                      @Divide by 16 by right shifting by four
    LDMFD R13!, {R8-R10, PC}        @Pop used registers and return to mainline

.data
Celsius_Temp:       .byte 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF
Celsius_Av:         .byte 0x0
FahrenheitTemp:     .byte 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
Fahrenheit_Av:      .byte 0x0
.align 2
STACK:  .rept 256
        .byte 0x00
        .endr

.end
```