```
// Assignment5 Project1:   Use buttons and LEDs
// Project1.s   contains main function

// Discovery board LED numbers
            .equ        RED,   0            // Red LED on PB6 = LED #0
            .equ        BLUE,  1            // Blue      LED on PB7 = LED #1
            .equ        ORANGE,      2            // Orange    LED on PB8 = LED #2
            .equ        GREEN, 3            // Green     LED on PB9 = LED #3

            .syntax unified
            .section    .text.ButtonLED
            .global main

// Delay  -  do nothing for N half-seconds
//   r0 = # half seconds
//   r1 modified

Delay:      ldr         r1, =0x0200000      // delay count for .5 seconds
Dloop1:             subs   r1, #1           // decrement delay count
                    bne         Dloop1              // repeat
                    subs   r0, #1           // # half seconds
                    bne         Delay               // repeat for each half second
                    bx          lr                  // return to main

// Main program
main:
                    bl          InitLEDs            // Initialize PB9-6 as outputs
to LEDs
                    bl          InitButton          // Initialize PA0 as input from
button

// Wait for button press to proceed
Top:        bl          CheckButton         // Check push button
                    cmp         r0, #1              // pressed?
                    bne         Top                 // no  -  repeat

                    mov         r4, #0              // Initialize count variable

// While button is held down
HoldButton:
                    // Count from 0 to 15 and display the number on LEDs
                    mov         r1, r4              // Move count variable to r1
                    mov         r0, RED             // Start with RED LED
CountLoop:  push   {r1}
                    and         r1, #1              // Get the LSB
                    bl          LED_OffOn           // Set LED state based on LSB
                    pop         {r1}
                    lsr         r1, #1              // Shift right for the next LED
                    add         r0, #1              // Move to the next LED
                    cmp         r0, #4              // Check if we've reached the
end of LEDs
                    blt         CountLoop           // Repeat if not

                    mov         r0, #1              // .5 second
                    bl          Delay               // delay
```

```
                    // Increment count
                    add         r4, #1
                    cmp         r4, #16                 // Check if count has
reached 16
                    bne         NoReset                 // Skip reset if not
                    mov         r4, #0              // Reset count
NoReset:
                    bl          CheckButton         // Check button state
                    cmp         r0, #1              // Check if button is
                    beq         HoldButton          // If yes, go back to HoldButton
loop

// Wait for the button to be pressed again
SecondPress:
                    bl          CheckButton         // Check push button
                    cmp         r0, #1              // pressed?
                    bne         SecondPress         // no  -  repeat

// While button is held down
HoldButton2:
                    mov         r0, GREEN           // Green LED
                    mov         r1, #1              // On
                    bl          LED_OffOn
                    mov         r0, #1              // .5 second
                    bl          Delay               // delay
                    mov         r0, GREEN           // Green LED
                    mov         r1, #0              // Off
                    bl          LED_OffOn

                    mov         r0, ORANGE          // Orange LED
                    mov         r1, #1              // On
                    bl          LED_OffOn
                    mov         r0, #1              // .5 second
                    bl          Delay               // delay
                    mov         r0, ORANGE          // Orange LED
                    mov         r1, #0              // Off
                    bl          LED_OffOn

                    mov         r0, BLUE            // Blue LED
                    mov         r1, #1              // On
                    bl          LED_OffOn
                    mov         r0, #1              // .5 second
                    bl          Delay               // delay
                    mov         r0, BLUE            // Blue LED
                    mov         r1, #0              // Off
                    bl          LED_OffOn

                    mov         r0, RED                 // Red LED
                    mov         r1, #1              // On
                    bl          LED_OffOn
                    mov         r0, #1              // .5 second
                    bl          Delay               // delay
                    mov         r0, RED                 // Red LED
                    mov         r1, #0              // Off
```

```
                    bl          LED_OffOn

                    bl          CheckButton         // Check button state
                    cmp         r0, #1              // Check if button is still
pressed
                    beq         HoldButton2         // If yes, go back to
HoldButton2 loop

                    b           Top                         // Go back to the start
```

LED_Driver.s

```
// Functions for LEDs on PB9-6

    .include "Equates.s"           // peripheral addresses

// Functions in this file
    .global InitLEDs              // init GPIOB9-6 for LEDs
    .global LED_OffOn             // individual LED OFF/ON
    .global DisplayCount          // display 4-bit # on LEDs

// Global variables defined in main file

    .syntax unified
    .section .text.LEDdrivers

// GPIOB initialization for LEDs: PB9-8-7-6
InitLEDs:
        // enable clock to GPIOB
        ldr         r0, =RCC
        ldr         r1, [r0, #AHBENR]
        orr         r1, #GPIOBEN
        str         r1, [r0, #AHBENR]
        // configure PB9-6 as output pins
        ldr         r0, =GPIOB
        ldr         r1, [r0, #MODER]
        bic         r1, #0x000FF000
        orr         r1, #0x00055000
        str         r1, [r0, #MODER]
        // set initial output values to 0
        ldr         r1, [r0, #ODR]
        bic         r1, #0x03C0
        str         r1, [r0, #ODR]
        bx         lr

// r0 = bit for LED# 3-0, corresponds to PB9-6
// r1 = 0 for off, 1 for on
LED_OffOn:
        push    {r0-r4}
```

```
        add         r0, #6            // change 3:0 to 9:6 for PB9-6
        mov         r4, #1            // on value
        lsl         r4, r4, r0        // shift 1 to position in 9:6
        ldr         r2, =GPIOB        // GPIO port B
        ldrh    r3, [r2, #ODR]    // read current ODR value
        bic         r3, r4            // clear bit for PBx
        cmp         r1, #1            // ON?
        bne         L1                // skip if ON
        orr         r3, r4            // set bit for PBx
L1:        strh     r3, [r2, #ODR]    // write new ODR value
        pop         {r0-r4}
        bx         lr                // return

// Display 4-bit Count on LEDs
// r0 = Count value (0-15)
DisplayCount:
        push    {r0-r4, lr}
        mov         r1, #0
        mov         r2, #GREEN
DisplayLoop:
        mov         r4, r0
        and         r4, #1
        bl         LED_OffOn
        lsr         r0, r0, #1
        add         r2, r2, #1
        add         r1, r1, #1
        cmp         r1, #4
        blt         DisplayLoop
        pop         {r0-r4, lr}
        bx         lr
```

Button_Drivers.s

```
// Functions for LEDs on PB9-6 and input button on PA0

            .include "Equates.s"          // peripheral addresses

// Functions in this file
            .global InitButton            // initialize PA0
            .global Init_EXTI0            // init button as EXTI0
            .global CheckButton           // return button state

// Global variables defined in main file

            .syntax unified
            .section .text.ButtonDriver

// GPIO initialization for button
InitButton:
            ldr         r0, =RCC                        // RCC register block
            ldr         r1, [r0,#AHBENR]          // read RCC_AHB1ENR
            orr         r1, #GPIOAEN              // enable GPIOA clock
            str         r1, [r0, #AHBENR]         // update AHB1ENR
            ldr         r0, =GPIOA                      // GPIOA register block
            ldr         r1, [r0, #MODER]          // current mode register
            bic         r1, #0x03
            str         r1, [r0, #MODER]          // update mode register
            bx          lr

// CheckButton            - return state of push button
// r0 = return value of 0 or 1
CheckButton:
            ldr         r0, =GPIOA                      // GPIO port A
            ldrh   r0, [r0, #IDR]                 // set bit
            and         r0, #0x01                       // mask all but bit 6
            bx          r14                                   // return
```