```
; Evan Heinrich
; CE2801 sect. 011
; 9/21/2021
;
; File: main.S
; Description of File:
;     Make one LED shift left across LED bar, then back right
; (opt) Dependencies: N/A

; Assembler Directives
.syntax unified
.cpu cortex-m4
.thumb
.section .text

; Constants
.equ RCC_BASE, 0x40023800
.equ RCC_AHB1ENR, 0x30        ; Offset from RCC_BASE
.equ GPIOBEN, 1<<1            ; GPIOBEN lives on bit 1 of AHB1ENR so shift a 1 left by 1

.equ GPIOB_BASE, 0x40020400
.equ GPIOB_MODER, 0x00        ; Offset from GPIOB_BASE
.equ GPIOB_ODR, 0x14          ; Offset from GPIOB_BASE

.equ SKIP_PB11, 0x800         ; Mask to determine if PB11 is the upcoming shift
.equ LEFT_MAX, 0x8000         ; Mask to determine if the leftmost LED is lit
.equ RIGHT_MAX, 0x20          ; Mask to determine if the rightmost LED is lit
.equ DELAY, 0x00034000        ; Delay
; Algorithm
;
;     Enable RCC For GPIOB
;     Set GPIOB as outputs
;     light the first light
;           loopLeft
;                 shift the light left
;                 determine if we need to skip PB11 yet (pattern recognition)
;                 branch to double-shift if we need to skip PB11
;                 delay
;                 compare to pattern for light all the way left active
;                 bne loopLeft
;                 beq loopRight
```

```
;           loopRight
;                 shift light right
;                 determine if we need to skip PB11 yet (pattern recognition)
;                 branch to double-shift if we need to skip PB11
;                 delay
;                 compare to pattern for light all the way to the right active
;                 bne loopRight
;                 beq loopLeft

; R1 = Address
; R2 = Scratch
; R3 = Masks
; R4 = Delay

; Expose main to the assembler
.global main

main:
      ; Enable RCC For GPIOB
      LDR R1, =RCC_BASE             ; Load the RCC base address
      LDR R2, [R1, #RCC_AHB1ENR]    ; Load what is currently stored in the AHB1 Enabler
      ORR R2, R2, #GPIOBEN          ; Apply the mask to enable GPIOB
      STR R2, [R1, #RCC_AHB1ENR]    ; Write back the new AHB1 Enabler value

      ; Set GPIOB Pins as Output
      LDR R1, =GPIOB_BASE           ; Load base address
      LDR R2, [R1, #GPIOB_MODER]    ; Load the GPIOB mode status
      LDR R3, =0xFF3FFC00                 ; Load the output clearing mask
      BIC R2, R2, R3                      ; Clear the modes
      LDR R3, =0x55155400                 ; Load the output setting mask
      ORR R2, R2, R3                      ; Overwrite with output set mask
      STR R2, [R1, #GPIOB_MODER]    ; Write back to memory


      ; Light the first light
      MOV R3, #0x20                       ; First Light Mask
      LDR R1, =GPIOB_BASE                 ; Load GPIOB Base Address
      LDR R2, [R1, #GPIOB_ODR]      ; Load the contents of the ODR
      ORR R2, R2, R3                      ; Apply First Light Mask
      STR R2, [R1, #GPIOB_ODR]      ; Store modified pattern
```

```
loopLeft:
      LDR R4, =DELAY          ; Prep delay
      1:                      ; Delay Loop
            SUBS R4, R4, #1
            BNE 1b

      LDR R2, [R1, #GPIOB_ODR]      ; Load the Current ODR
      LSL R2, R2, #1                ; Logical Shift Left
      CMP R2, #SKIP_PB11            ; Compare to see if PB11 would be active
      BEQ doubleShiftLeft           ; If PB11 is the next shift, shift again
      STR R2, [R1, #GPIOB_ODR]      ; Store shifted value

      leftBranch:
            CMP R2, #LEFT_MAX ; Determine if we will continue looping
            BNE loopLeft      ; Repeat loop if we have not shifted left all the way
            BEQ loopRight     ; Start shifting right if we are done shifting left
      doubleShiftLeft:
            LSL R2, R2, #1                ; Shift to the left again
            STR R2, [R1, #GPIOB_ODR]      ; Store shifted value
            B loopLeft                    ; Return to the LSL loop

loopRight:
      LDR R4, =DELAY          ; Prep Delay
      1:                      ; Delay loop
            SUBS R4, R4, #1
            BNE 1b

      LDR R2, [R1, #GPIOB_ODR]      ; Load the current ODR
      LSR R2, R2, #1                ; Logical Shift Right
      CMP R2, #SKIP_PB11            ; Compare to see if PB11 would be active
      BEQ doubleShiftRight          ; Jump to the double-shift location for LSR
      STR R2, [R1, #GPIOB_ODR]      ; Store shifted value

      rightBranch:
            CMP R2, #RIGHT_MAX ; Determine if we will continue looping
            BNE loopRight      ; Loop if we haven't shifted all the way to the right
            BEQ loopLeft       ; Start shifting left

      doubleShiftRight:
            LSR R2, R2, #1                ; Shift right one more time
            STR R2, [R1, #GPIOB_ODR]      ; Store shifted value
            B loopRight                   ; Return to LSR loop
```